Summary

This report documents the main stages taken and used to help create an Information
leagues at an indoor sports centre.

It was intended that the system would be created, for Mr C Larkin, at the Bentley Park
objectives of the system were clearly outlined as a foundation for the rest of the
outlined below:

- Use the database to maintain scores and league tables, improving the current pro
The centre holds many leagues running simultaneously, on most nights of the week. Each le
and a set of fixtures for these leagues. The system created should be able to tak
update a league table, so the user can keep track of the leagues. The system should aim t
current procedures by providing easier, quicker and by meeting all user requirements.

- Investigate making data in the system available on the Web and any security issu
The centre may benefit by having information about leagues available on the Web fo
is not an objective to implement this, but to research the possibility of this hap
would entail.

On completion of the project many things were achieved. A complete working system wa
to meet the needs of the user, which incorporated good usability and functionality
were made to the system where it was felt that it would better meet the defined requirements
user, but helped in the overall performance of the system.

In addition to the system that was created, carrying out the project helped me gai
different subject areas studied during the project. The deliverables on completion of the pro

- Database System
- Demonstration of system to user
- This project report

The main stages of the project are detailed in the rest of this report.

Acknowledgements

I would like to acknowledge the following people on completion of this project.

- Dr Kevin McEvoy, my project supervisor, for his help and guidance throughout the
- Mr C Larkin for his input into how the system would best meet his needs.

Contents

I – Summary i
II – Acknowledgements ii
III – Contents iii
Section – 1 Introduction 1
1.1) Background 1
1.2) Overview of the Problem 1
1.3) Scope of the Project 1

Section – 2 Background Research & Analysis 2
2.1) Database Management Systems 2
2.1.-1 Microsoft Access 3
2.1.-2 Microsoft SQL Server 3
2.1.-3 Oracle RDBMS 3
2.1.-4 Conclusions for Database Management Systems 4
2.2) Web Based Front End 4
2.2.-1 HTML 4
2.2.-2 Dynamic HTML 4
2.2.-3 ActiveX Data Objects 5
2.2.-3 Conclusions for Web Based Front End 5
2.3) Server Side Processing 5
2.3.-1 Active Server Pages 5
2.3.-2 Data Access Pages 6
2.3.-3 Conclusions for Server Side Processing 7
2.4) Review of Existing Systems 7
2.4.-1 Fixture Generation/ League Management Software 7
2.4.-2 Conclusions 7
2.5) User Needs 8
2.5.-1 User Needs Essential to the System 8
2.5.-2 User Needs that may possibly be implemented 8
2.6) Requirements of the System 9

Section – 3 Design of the System 11
3.1) Data Modelling 11
3.1.-1 Entities 11
3.1.-2 Attributes 11
3.1.-3 Relationships 12

3.1.4 ER Diagram 13
3.2) Database Design 14
3.2.1 Database Schema (table structure) 14
3.2.2 Data Definition 15
3.3) Normalisation 16
3.3.1 Definition of a superkey, key and various normal forms 16
3.3.2 Normalisation of Data into 1NF 17
3.3.3 Normalisation of Data into 2NF 19
3.3.4 Normalisation of Data into 3NF 19
3.3.5 Normalisation of Data into BCNF 20
3.4) Integrity Constraints and Business Rules 21
3.4.1 Validation Rules 21
3.4.2 Integrity Constraints 21
3.4.3 Integrity Rules 21
3.4.4 Business Rules 23
3.5) Query Design 24
3.6) Form Design 24
3.6.1 Form Hierarchy 26
3.7) Human Computer Interaction 26
3.8) Web Based Front End 27

Section-4 Implementation of the System 28
4.1) Implementation of the Database 28
4.1.1 Implementing Tables 28
4.1.2 Enforcing Relationships 28
4.1.3 Implementing Database Constraints 30
4.1.4 Query Implementation 30
4.1.5 Fixture Generation 32
4.1.6 Managing Fixtures/Results 33
4.1.7 Maintaining League Tables 34
4.1.8 Implementation of Forms 35
4.1.9 Implementation of Reports 40
4.1.10 Macros 40
4.2) Implementation of Web Based Front End 41

Section-5 System Testing 43
5.1) Black Box Testing 43
5.2) White Box Testing 45

5.3) Conclusions of Testing 46

Section-6 Demonstration and Feedback 47
6.1) Details of Demonstration 47
6.2) Feedback from Demonstration 47

Section-7 Evaluation and Future Developments 48
7.1) Evaluation of the System 48
7.2) Future Enhancements 49
7.3) Conclusions 50

References 51

Appendix A Project Experience 52

Appendix B Relationship Diagram of Tables 53

Appendix C Add Week Function 54

Appendix D Example Reports 55

Appendix E ASP Files 58

Appendix F Web Page Screen-Shots 62

Section 1 - Introduction

1.1) Background

Benyon Park is an indoor football centre on the outskirts of Leeds City centre. I
opened just over a year ago and knowing the owners of the centre, I was intereste
documentation of results and league tables were handled. Having looked at the sys
the centre I spoke with Mr. C Larkin and said that the current system could be imp
improvements I would make to the system, it was decided that this project would bene

1.2) Overview of the Problem

From the project objectives we know that a system will have to be created that al
fixtures and update the league table. To begin with the existing system will h
and then a plan will be produced to establish how current procedures can be improv
User Needs. Along with the user needs a set of possible enhancements should also be discussed and agree
the end user. Background research will then look into available programs, similar
implement, and also what available tools are available to aid in the design and implementation of
form Section 2 of the report.

Because this is a database project, a sound database should be designed, concentra
normalisation, integrity checks and also the design of the information provided to the user w
usability in mind. If the database is designed concisely the process of implement
the possibility of finding errors later in the project. Design and Implementation
and 4 of the report respectively. Once the system has been created it should be subjected t
before demonstrating to the end user.

1.3) Scope of the Project

The system should be developed to a high standard and should be created in order to ensure
system capable of creating and managing football leagues running at Benyon Park.
system in a generic nature so it could be possible to use in other centres, if and

Section-2 Background Research & Analysis

This section outlines the background research carried out in order to assess how t
implemented. Various tools and technologies have been looked at which could be us
needs and system requirements have been established.

2.1) Database Management Systems

“A DBMS (database management system) is a program that lets computer users create
database” ‘(Mott & Roberts, 1998)’. The DBMS manages user requests (programs, users
that they have no need to understand where the data is physically located on stora
multi-user, the DBMS hides from each user, whoever else may also be accessing the da

In handling user requests, the DBMS ensures the integrity and security of the data. It makes s
constantly accessible, consistently organised as intended and that only those with
retrieve the data. The most typical DBMS is a relational database management system. A newer kind of DBM
is the object-oriented database management system.

A DBMS can be thought of as a file manager that manages data in a database. A database is a collection of d
that is organised so that its contents can be accessed, managed and updated. There are a var
different databases that can be implemented, including a relational database, a di
orientated database. The most prevalent type of database is the relational database, which is a tabular database
which data is defined so that it can be reorganised and accessed in a number of di

A relational database is a set of tables containing data that is fitted into predet
contains one or more data categories in columns. Each row contains a unique instance o
defined by the columns. The standard user and application program interface to a
structured query language. SQL statements are used for interactive queries for obtaining informa
relational database, and also for gathering data for reports.

A relational database would be an ideal way to implement the system for Benyon Par
to create and access, as well as having the advantage of being easy to extend. After the
a new data category can be added without requiring all of the existing applicatio

A DBMS is usually an inherent part of a database product. Because the relational database management
(RDBMS) family includes such a large number of products, I have focused on the mos
Microsoft Access, Oracle and SQL Server.

2.1.1 Microsoft Access

Microsoft Access is an example of a single user DBMS and one of the most well know
implementations of the relational data model. It is considered to be part of an i
Windows platform, for creating and managing databases. Access is a database engine and a graph
interface (GUI) for data definition and manipulation through SQL. The programming
also available to the users of Access. Users can create queries, forms and reports
available Wizards, which can provide for input and output operations against the d
interactive programs that direct the user through a set of questions to help them
to accomplish.

Access is a RDBMS, which comprises several components. The Microsoft Jet Engine i
managing the data. The engine stores all the application data, such as tables, fo
modules. The engine also provides advanced capabilities such as various types of data access. T
calls the engine to provide various services, such as the storage and retrieval of

2.1.2 Microsoft SQL Server

Microsoft's SQL Server is an example of a DBMS that serves multiple client users
very easy database server to use and administer, offering high levels of automation, full
graphical tools. Many activities can be performed in several ways, including
text-based interfaces.

SQL Server provides several wizards that assist the user in completing major activ
performance, security and object management. Graphical tools are provided which s
the database and the query analyser suggests indexes that can be used for the best

2.1.3 Oracle RDBMS

Oracle is currently the more widely used Relational Database Management System. A
an Oracle Database and the Oracle Instance. The Oracle Database is the collection of
log and control files. The Oracle Instance is the Oracle system processes and the
created for a specific instance of the database. SQL is supported by the Oracle server
define and manipulate data. PL/SQL is the procedural language that controls the
variables and provides looping procedures. Oracle can be accessed through languages such
as C or Java.

2.1.4 Conclusions for Database Management Systems

Although each of these Relational Database Management Systems discussed would help system, I have decided to use Microsoft Access. I have used this software a number of times in the with a variety of projects undertaken. I feel comfortable that I can produce the best quality system using the knowledge that I have, along with new techniques that I will learn.

2.2) Web Based Front End

If I were to implement the feature of being able to view information stored in the need to know how to display the information on the browser for the user. I have r dynamics HTML and ActiveX Data Objects, which will help in retrieving and displaying the

2.2.1 HTML

HTML (Hypertext Mark-up Language) is the set of mark-up symbols that are inserted into a file tha for display on a World Wide Web browser page. The mark-up tells the Web browser exactly how page's words and images should be displayed for the user. Each individual mark-up code is referred to as a element or tag. Some tags come in pairs, which are used to indicate when a particular and end, an example being font colour.

HTML is recommended by the World Wide Web Consortium and is adhered to by major br Microsoft Internet Explorer and Netscape Navigator. Both browsers support HTML 4 implement some more advanced features of HTML 4.0 differently, so a web developer for both browsers and send out the suitable version to the user.

2.2.2 Dynamic HTML

Dynamic HTML describes a set of new HTML tags and properties that provide the Web developer with the capabilities to create a more animated Web page. It can also make the Web page mo interaction than previous versions of HTML allowed. Examples of unique features t provide, are where the colour of a text heading changes when the mouse passes over it, an is able to 'drag and drop' an image to another place on a Web page. These feature Web documents to look and act like desktop application

The main problem to overcome when using dynamic HTML is that many users still use Therefore a Web developer must create two versions of each site, so that the pages

browser can be returned. Dynamic HTML is more powerful programming in Web pages, purely because it can address more elements of a page.

2.2.3 ActiveX Data Objects

ActiveX Data Objects (ADO) is an application program interface from Microsoft that is used in writing Windows applications to gain access to a relational or nonrelational database that is from Microsoft and other database providers.

ADO will allow a user to write a program that would provide users of a Web site with access to a database (e.g. an Oracle or Access database). This would prove extremely useful if some of the data from the system I will produce is to be made available over the Web. ADO programs are written in an HTML file, which is saved and recognised as an Active Server Page. When a user requests the page from the Web site, the page sent back would include any appropriate data from a database, or

2.2.4 Conclusions for Web Based Front End

Because the Web pages will be nothing more than a way of viewing certain information and the user will have no need or indeed no possibility to make alterations, I have decided to use HTML as the language in HTML. This is because it is supported by the major browsers and is programmable because it will be

2.3) Server Side Processing

Again, if I were to implement the feature of being able to view information stored in tables and queries then I will have to know how to make the data from tables and queries accessible to the user. I researched into ways of solving this problem through the use of Active Server Pages and Remote Data Access Pages.

2.3.1 Active Server Pages

Active Server Pages are server-generated pages, which can call other programs for a variety of reasons like accessing databases and serve different pages to different browsers. These can be pages that include embedded programs that are processed on a Microsoft Web server before they are sent to the user. ASP is similar to a common gateway interface application, in that they both run on the server, usually customising a page for the user. Typically, the script uses input received as the result of the form to request the page to access data from a database and format the page before sending it to the requestor.

ASP is a feature of the Microsoft Internet Information Server - however, it is quite
building a regular HTML page can be delivered to almost any browser. This is a benefi
we can't predict what browser our range of users will be using. This will give a
the page.

An ASP file can be created by including a script written in VBScript or Jscript in an HTML file,
ActiveX Data Object program statements in the HTML file. Microsoft says that it is
ASP rather that the client, because the server can easily displayable HTML pag
whereas the client script may not work as intended on some of the older browsers.

2.3.2 Data Access Pages

A data access page is a special type of Web page designed for viewing and working
intranet. The data that is accessible from these Web pages are stored in a Micros
Microsoft SQL Server database. The data access page can also include data from o
Excel. Data access pages are created in Microsoft Access, however the page is a separat
in some outside location with a shortcut to the file provided in the Database win
similar to designing forms and reports, but with some significant differences.

Data access pages are connected directly to a database. When users display the da
Internet Explorer, they are viewing their own copy of the page. This means that a
to the way the data is displayed, affects only their copy of the page. However, any ch
data itself are stored in the underlying database and are available for everyone

The database that is the data source for a data access page must be on a shared server in order for
view the page in a Web browser. One possible solution to this would be to publish
Web servers. The HTML file corresponding to the data access page and any related folders in Windows
Explorer, could be copied to a folder underneath the root directory of the Web ser
should be made accessible to the chosen Web server.

A data access page is formed from a shortcut to an MS Access database and a corresponding HTML file
located on the computer's file system. There are two ways to protect a page's sho
being renamed, deleted, or changed, the computers file system security can be used.
On the computer, the Access database that contains the page's shortcut can be made
server the file and the folder where the HTML file is located. This will solve the
problem of any data data access page being altered by the users, where only data is
basis.

2.3.3 Conclusions for Server Side Processing

For a user to view and work with a Data Access Page on the Internet, they must hav
Internet Explorer 5 and a Microsoft Access 2000 license. This would be a major prob
have access to these.

Active Server Pages appear to be the best way to make information accessible on th
Data Objects introduced in section 2.3.2 created database can be accessed and relevant data
and returned to the browser.

2.4) Review of Existing Systems

There are many existing systems available that can be used to create fixtures for
they can also manage the league tables when results have been entered. Some of th
described below.

2.4.1 Fixture Generation/League Management Software

League Maker 2000 is a software package that helps create and run leagues. It creates leagues for a n
teams, helps customise league tables, previews league fixtures, prints league fixt
reschedule matches that can’t be played. The software runs on personal computers operating systems,
with any version of Windows. There are two relevant programs, one that produces t
other that produces fixture lists. A problem of this software is that creating fi
tables is carried out using separate programs. I intend that a single system should do b

SportSoft is another piece of software that has been designed to speed up and minimis
league tables and fixture lists. SportSoft Again comprises of two programs, one to create fixtures and another to
the league tables. The Competition Fixtures program creates fixtures for as many
needed, prints selected fixture lists and provides a matrix format for fixtures. The fixtures can
be exported to other software packages. SportSoft League Tables. This program creates and mainta
number of leagues, allows the user to customise their league table, prints tables
has an improves feature where statistics can be performed upon the results of the fi

2.4.2 Conclusions

Clearly, these systems have not been designed for the individual. Although the so
features required to produce for our needs, i.e. the fixture generation and league t
they don’t provide for every need. The existing systems also require data to be e

another. Instead of having separate programs to handle different tasks, the system aims to create wil
features in one system, a system that contains the means to produce fixture lists,
store contact details. This system will aim to provide for the individual user as

2.5) User Needs

A meeting was set up with Mr. C Larkin at a very early stage of the project in ord
needs of the system. To begin with my initial plans of what the system would do,
on the basis of these initial ideas a set of full user needs were produced that w
and implementation stages, which have been split into essential and possible user

2.5.1 User Needs Essential to the System

1.1 The system shall maintain up-to-date contact details for each team in each league.
1.2 The system shall maintain contact details for referees currently officiating th
1.3 The system shall maintain contact details for teams that are on the waiting lis
1.4 The system shall be able to print out fixture lists for any of the leagues runn
distributed on notice boards.
1.5 The system shall be able to print out results of fixtures after any particular league
distributed on notice boards.
1.6 The system shall be able to update and print out league tables, to be distributed on notice boa
1.7 The system shall allow teams to be added to the waiting list to be entered into
1.8 The system shall create fixtures for a set of given teams, for any given league
1.9 The system shall maintain league tables for each league.
1.10 The system shall allow for the results of each fixture to be entered into the d
1.11 The system shall take the results for each game in a league and automatically upd
1.12 The system shall renew or remove any league where all of the fixtures have been

2.5.2 User Needs that may possibly be implemented

2.1 The system shall keep track of when the football pitches are in use and when they a
2.2 The system shall take bookings for a particular football pitch at a particula
2.3 The system shall keep track of which referees are officiating for which matches they
charge of.
2.4 The system shall print out a score sheet for each referee, with details of al
of each night.

2.5 The system shall make league tables accessible on the web.
2.6 The system shall make fixture lists and results accessible on the web.
2.7 The system shall create newsletters containing results, league tables, forthc
reports, to be distributed amongst teams and on notice boards.

2.6) Requirements of the System

This is a detailed requirements specification for the intended product. The syste
been cross referenced with the essential user needs.

Information to be stored by the system

Teams in leagues and teams on waiting list
Team name, Contact name, Street name, City, County, Postcode, Contact number.

The information above will allow the system to carry out User Needs 1.1 and 1.3.

User Need 1.7 states that the system should allow for teams to be added to the wai
for those who use the system. This means the system will store the team details,
the waiting list and the day that the team wishes to play on.

Referees currently officiating the leagues
Referee name, Contact number, Street name, City, County, Postcode.

The information above will allow the system to carry out User Need 1.2.

Fixtures
Home team name, Away team name, FixtureID, Pitch number, Time of match, Date of ma

As we have the League name we are able to separate the fixtures so that they are ordered
leagues. This information will allow the system to carry out User Need 1.4.

User Need 1.8 states that the system should create fixtures for the system will
need the list of teams to have the fixtures created for, the date the league start
will be played. If a set of fixtures for a league has been completed and the user
renew the league, this will also allow the system to carry out User Need 1.12.

Results

Home team name, Goals scored by home team, Away team name, Goals scored by away te
of match, League name.

As we have the League Name we are able to separate so that they are ordered into th
leagues. This information will allow the system to carry out User Need 1.5.

User Need 1.10 states that the system should allow for results to be entered into
Need for those who use a system. This means that the system will store the goals sc
teams against the fixture. This will also allow the system to carry out User Nee

League Table

League name, Team names, Games played, by each team, Games won by each team, Games lost by each
Games drawn by each team, Goals scored by each team, Goals conceded by each team,
team, Total of points gained by each team.

As we have the League Name we are able to separate so that they are ordered into
leagues. This information will allow the system to carry out User Needs 1.6 and

Section-3 Design of the System

3.1) Data Modelling

Data modelling is the most important part of designing a database application. Important stages of data modelling are; identifying entities and attributes; identifying the relationships between the entities and finally using the entities, attributes and relationships to produce an entity-relationship diagram. The ER diagram can then be translated to the tables in the relational database.

3.1.1 Entities

An entity is a real-world object or concept that the ER model represents. An entity type is a collection of entities that have the same attributes. Here are the identified entity types that belong to the system:

1. Team
2. League
3. Fixture
4. Pitch
5. Day
6. Teams on Waiting List
7. Referee

3.1.2 Attributes

Each entity stated above has attributes associated with them, which are used to describe the entity. There are many kinds of attributes associated with the entities, examples of which are; attributes can be divided into smaller parts (e.g. address can be split into street, city, county, postcode), and key attributes; that give a distinct value for each individual entity in the set. Here is the preliminary list of the associated attributes in the ER model based on the system requirements (Section 2.1.6):

TEAM

Team ID, Team Name, Contact Name, Address (Street, City, County, Post Code), Contact Number, Plays In (League, Fixture)

LEAGUE

League ID, League Name, Day, Set Of (Teams)

FIXTURE

Fixture ID, Pitch Number, Match Time, Date, Teams (Home Team, Away Team), Result

PITCH

Pitch Number

DAY

Day ID, Name of Day

TEAMS WAITING

Team Name, Contact Name, Address (Street, City, County, Post Code), Contact Numbe
Waiting Date

REFEREE

Referee ID, Name, Contact Number, Address (Street, City, County, Post Code),
Works On (Days)

3.13 - Relationships

A relationship is any interaction that exists between the entities. Using the de
types could now be identified.

1. PARTICIPATING_TEAMS, a 1:N relationship type between LEAGUE and TEAMS. Both pa
are total.

2. WORK_ON , a M:N relationship type between REFEREE and DAY. Both participati
referee must be down to work for at least one day.

3. HAS_PREFERRED_DAY , a 1:N relationship type between DAY and TEAMS WAITING. Bo
participation are total.

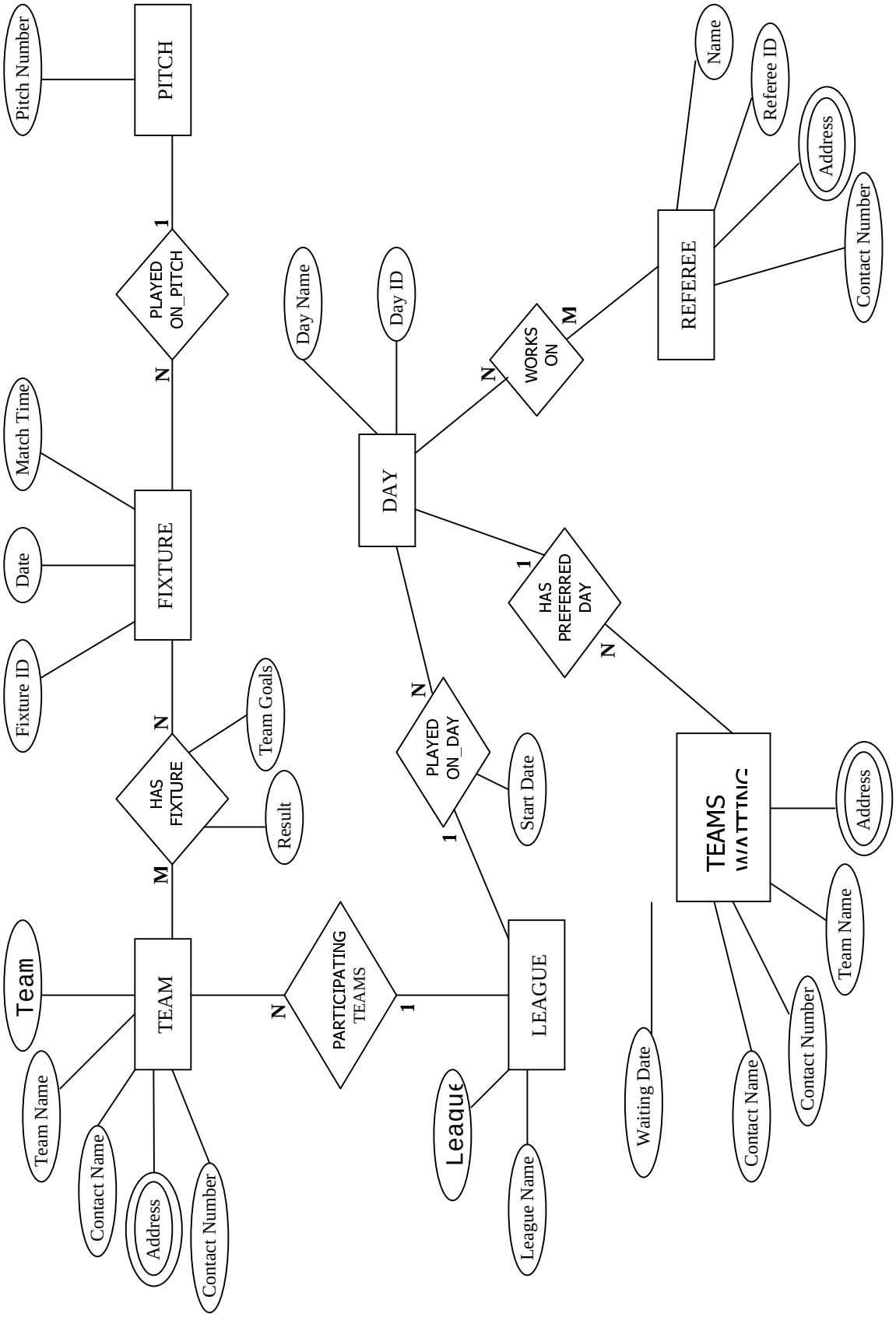
4. ON_DAY , a 1:N relationship type between DAY and LEAGUE. Both participations

5. ON_PITCH, a 1:N relationship type between PITCH and FIXTURE. Both participat
fixture cannot exist without a pitch to play on.

6. HAS_FIXTURE, a M:N relationship type between TEAM and FIXTURE. Both particip

All attributes that have been refined into the above relationships are now remove
preliminary design of the database. This will help remove redundancy, which is importa
design the conceptual schema of the database” (‘Elmasri & Navathe, 2000’). Some
the storage level, but can be introduced later. This may be needed for keeping
has won, lost or drawn, to keep a total.

3.1.– ER-Diagra


PITCH
Pitch Number
PLAYED ON_PITCH
1
N
FIXTURE
Match Time
Date
Fixture ID
HAS FIXTURE
N
M
Team Goals
Result
TEAM
Team
Team Name
Contact Name
Address
Contact Number
PARTICIPATING TEAMS
N
1
LEAGUE
League
League Name
DAY
Day Name
Day ID
PLAYED ON_DAY
N
1
Start Date
HAS PREFERRED DAY
1
N
WORKS ON
N
M
REFEREE
Name
Referee ID
Address
Contact Number
TEAMS WAITING
Waiting Date
Address
Team Name
Contact Name
Contact Number

3.2) Database Design

3.2.1 Database Schema (table structure)

The overall description of the database is the database schema. this is specified during the design shouldn't change, unless requirements of the database applications change. Shown for the relational database schema:

TEAM

TEA	TEAM NAME	LEAGUE ID	CONTACT NAME	STREET NAME	CITY	COUNTY	POST CODE	CONTACT NUMBER

LEAGUE

LEAGU	LEAGUE NAME	DAY

TEAM_FIXS

TEAM	FIXTU	GOAL S SCORED	VENUE	WON	LOST	DRAWN

FIXTURES

FIXTU	PITCH_ID	MATCH TIME	DATE

PITCH

PITCH

REFEREE

REFER	REFEREE NAME	CONTACT NUMBER	STREET NAME	CITY	COUNTY	POST CODE

REF_DAY

REFER	DAY

DAY

DAY	DAY NAME

TEAMS WAITING

TEAM	CONTACT NAME	STREET NAME	CITY	COUNTY	POST CODE	CONTACT NUMBER	PREFERRED DAY	WAITING DATE

3.3.2 Data Definition

Table Name	Attribute Name (key attributes in italics)	Attribute Type/ Length	Null Values
TEAM	*Team_ID*	AutoNumber	Not Null
	Team_Name	Text (20)	Not Null
	League_ID	Number (long integer)	Not Null
	Contact_Name	Text (40)	Not Null
	Street_Name	Text (50)	Null
	City	Text (30)	Null
	County	Text (20)	Null
	Postcode	Text (9)	Null
	Contact_Number	Text (15)	Not Null
LEAGUE	*League_ID*	Number (long integer)	Not Null
	League_Name	Text (40)	Not Null
	Day	Number (integer)	Not Null
TEAM_FIXS	*Team_ID*	Number (long integer)	Not Null
	Fixture_ID	Number (long integer)	Not Null
	Goals_Scored	Number (integer)	Null
	Venue	Text (2)	Not Null
	Won	Number (integer)	Null
	Lost	Number (integer)	Null
	Drawn	Number (integer)	Null
FIXTURES	*Fixture_ID*	Number (long integer)	Not Null
	Pitch_ID	Number (integer)	Not Null
	Match_Time	Number (long integer)	Not Null
	Date	Date/Time (shorted date)	Not Null
PITCH	*Pitch_Number*	Number (integer)	Not Null
REFEREE	*Referee_ID*	AutoNumber	Not Null
	Referee_Name	Text (40)	Not Null
	Contact_Number	Text (15)	Not Null
	Street_Name	Text (50)	Null
	City	Text (30)	Null
	County	Text (20)	Null
	Postcode	Text (9)	Null
REF_DAY	*Referee_ID*	Number (long integer)	Not Null
	Day_ID	Number (integer)	Not Null
DAY	*Day_ID*	AutoNumber	Not Null
	Day_Name	Text (10)	Not Null

TEAMS WAITING	Team Name	Text (20)	Not Null
	Contact Name	Text (40)	Not Null
	Street Name	Text (50)	Null
	City	Text (30)	Null
	County	Text (20)	Null
	Postcode	Text (9)	Null
	Contact Number	Text (15)	Not Null
	Preferred Day	Number (integer)	Not Null
	Waiting Date	Date/Time (short date)	Not Null

3.3) Normalisation

Normalisation of data is the "process of analysing the given relation schemas based on their functional dependencies and primary keys to minimise redundancy and minimise the insertion, deletion and update anomalies" ('Elmasri & Navathe, 2001'). The goal of relational database design is to have relations that if relations are in normalised form then there will be minimum data redundancy and minimum chance for something to go wrong. The normal form of a relation refers to the highest normal form condition that it meets, and hence indicates to what extent it has been normalised.

Normalisation must confirm the existence of two properties that the relational schema R possesses. These properties are as follows.

1. The lossless join/non-additive join property which guarantees that the spurious tuple generation problem doesn't occur with respect to the relational schemas created after decomposition.
2. The dependency preservation property, which ensures that each functional dependency is represented in some individual relations resulting after decompo

3.3.1 Definitions of a superkey, key and various normal forms

Superkey – The superkey of relation schema $R = \{A_1, A_2, \ldots, A_n\}$ is a set of attributes $S \subseteq R$ with the property that no two tuples t_1 and t_2 in any legal relation for R will have $t_1[S] = t_2[S]$.

Key – Any key K is a superkey with the additional property that removal of any attribute from K will cause K not to be a superkey anymore.

BCNF (Boyce-Codd Normal Form) – A relation R is in BCNF if and only if, whenever there exists a functional dependency $A_1, A_2, \ldots A_n \rightarrow B$ for R, it is the case that $\{A_1, A_2, \ldots, A_n\}$ is a superkey for R.

1st Normal Form – Any relation with atomic values in each of the relation cells is in 1NF. i.e. no multi-valued attributes.

2nd Normal Form – For a relation to be in 2NF then it must be in 1NF. There must not exist any non-trivial dependencies $A_1 \ldots A_n \rightarrow B$ such that the left hand side is a proper subset of a key, unless B is a member of some key.
(All BCNF relations are in 2NF, but not all 2NF relations are in BCNF).

3rd Normal Form – A relation R is in 3NF if and only if, where a non-trivial dependency $A_1, \ldots, A_n \rightarrow B$ for R, it is the case that A is either a superkey, or B is a member of some key.
(All BCNF are in 3NF)

3.3.2 Normalisation of Data into 1NF

To be in 1NF the relation must only have single atomic values. Consider the LEAGUE relation schema whose primary key is LEAGUE_ID, and suppose we extend it by including the TEAM_ID attribute. We assume that each league can have a number of teams.

LEAGUE

LEAGUE	LEAGUE NAME	DAY	TEAM

This is not in 1NF because TEAM_ID is not an atomic attribute. The TEAM_ID domain contains atomic values, but some tuples can have a set of these values. In this case, TEAM_ID is functionally dependant on LEAGUE_ID. The domain of TEAM_ID contains sets of values and hence is non-atomic. In this case LEAGUE_ID is functionally dependant on TEAM_ID because each set is considered a single member of the attribute domain. In either case this relation does not satisfy 1NF, and does not qualify as a relation.

LEAGUE	LEAGUE NAME	DAY	TEAM_ID
4	Social League	1	{MOS, Brazil Nuts, Skylark}
5	Mechanics League	3	{Troopers, Magic, De Puy}

Achieving 1NF

With a problem like this there are three ways to achieve 1NF:

1. Remove attribute TEAM_ID that violates 1NF and place it in a separate relation along with the primary key LEAGUE_ID. The primary key for the new relation is {TEAM_ID} and the foreign key is {LEAGUE_ID}. This decomposes the non-1NF relation into 2 1NF relations.
2. Expand the key so that there will be a separate tuple in the original relation for each member of the primary key now becomes a combination of {LEAGUE_ID, TEAM_ID}. The disadvantage of this is introduction of redundancy.

LEAGUE	LEAGUE NAME	DAY	TEAM_ID
4	Social League	1	{MOS}
4	Social League	1	{Brazil Nuts}
4	Social League	1	{Skylark}
5	Mechanics League	3	{Troopers}
5	Mechanics League	3	{Magic}
5	Mechanics League	3	{De Puy}

3. If a maximum number of values are known, we can replace the TEAM_ID with a number of atomic attributes TEAM_ID1, TEAM_ID2, ..., etc. The disadvantage of this method is introducing null values if some of the leagues have fewer teams.

The first of these three methods is the superior as it introduces no redundancy and is the method we have already used in our relational schema as we have two separate relations for LEAGUE and TEAM.

The first normal form also disallows nested relations or relation-valued attributes. This means that multi-valued attributes themselves are not allowed. If nesting was allowed this is how it might look.

FIXTURES

	Results						PITCH_ID	MATCH TIME	DATE
	TEAM_	GOALS SCORED	VENUE	WON	LOST	DRAWN			
1	2	4	H	0	1	0	4	1900	29/3/01
	3	5	A	1	0	0			
2	4	7	H	1	0	0	1	1830	2/4/01
	6	2	A	0	1	0			

Each tuple represents a fixture entity and a relation ‘results’ represents the team. To normalise this into 1NF, we remove the nested relation attributes and PLUS them into

and include the primary key from the FIXTURES relation. Decomposition and primary key propagat
the FIXTURES and TEAM_FIXS schemas as shown in the database schema (Section 3.2.1).

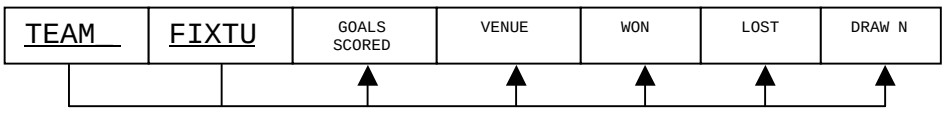
3.3.3 Normalisation of Data into 2NF

This is based on the concept of full functional dependency. X → Y is a full functional dependency if the
of an attribute A that exists in X results in the dependency not holding. X → Y is a partial dependency
some attribute A that exists in X can be removed from X and the dependency still holds.

The test for 2NF involves testing for functional dependencies, where the left hand
part of the primary key of the relation. A relation schema R is in 2NF if every
is fully functionally dependant on the primary key of R.

Most relations in the relation schema have a primary key made up of a single attri
nonprime attribute in the relation is fully functionally dependant on the primary key. TEAM_FIXS is in
the primary key is made up of {TEAM_ID, FIXTURE_ID} combinations. If either TEAM_ID or
FIXTURE_ID attribute were removed from the left hand side of the dependencies with
attributes, then the dependencies would hold. This means that each nonprime attri
fully functionally dependant on the primary key.

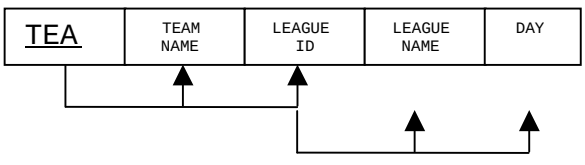
TEAM_FIXS



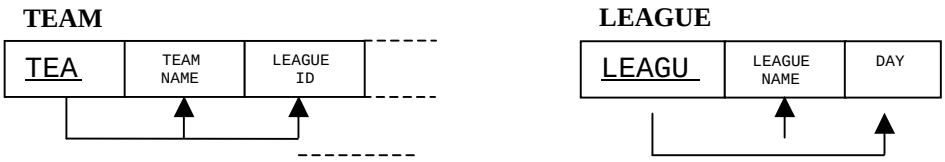
3.3.4 Normalisation of Data into 3NF

Third normal form is based on the concept of transitive dependency. X → Y is a transitive dependency if t
a set of attributes Z that is neither a candidate key nor a subset of any key of R, and both X → Z and Z → Y hold.
According to Codd's original definition, a relation schema R is in 3NF if it is in 2NF and no nonprimeatis
attribute of R is transitively dependent on the primary key. If we had the follow

TEAM



The dependency TEAM→LEAGUE_NAME is transitive through LEAGUE_ID because the depen
TEAM_ID→LEAGUE_ID and LEAGUE_ID→LEAGUE_NAME both hold, and LEAGUE_ID is neither a
key nor a subset of the key of TEAMS. We can normalise this relation by decompos
relations:



These are the relations that we have in the schema, which is the 3NF which shows that there are no
dependencies in the schema and that we have achieved 3NF.

3.3.5 Normalisation of Data into BCNF

“Boyce-Codd Normal Form was proposed as a simpler form of 3NF, but was found to be
because every relation in BCNF is also in 3NF; however, a relation in 3NF is not
& Navathe, 2000’).

It is easy to check that a relation from TEAM_STATS is already in BCNF. Take one for example. The only non
dependency is

TEAM_ID,FIXTURE_ID → GOALS_SCORED ,VENUE ,WON ,LOST,DRAWN

The relation is in BCNF because {TEAM_ID, FIXTURE_ID} is a key and therefore a superkey. The
for every other trivial dependency in the relation schema, which means that the schema ho

3.4) Integrity Constraints and Business Rules

This part of the system design is based on database integrity. We want to ensure that the data w database is consistent and within the limits set in the database design. This will ensure a complete and accurate representation of the Universe of Discourse. There are se used to meet these goals, which allow the designer to restrict the values that can be entered into integrity rules, integrity constraints and business rules are terms that describe

3.4.1 Validation Rules

Validation rules specify requirements for data entered into a record, field or con structure, i.e. text box, placed on a form for the purpose of data entry). These a

- The Postcode field in the Teams Waiting table must be in the format ‘LL09
- The Contact_Number field in the Teams Waiting table must be in the format ‘(00 000000’.
- The Preferred_Day and Day fields in the Teams Waiting table must be an integer valu between 1 and 7.
- The Venue field in the Teams Waiting table must be in the format of either “H” o
- The Pitch field in the Teams Waiting table must be an integer value between 1 and 4 as there

3.4.2 Integrity Constraints

These are defined through the Data Definition Language (DDL) and prevent any data that violate constraints, from being entered into the database, no matter how the data entry is

3.4.3 Integrity Rules

The relational model has two integrity rules, the entity integrity rule and the re outlined below:

1. The entity integrity rules

The only condition for these rules is that no part of the primary key should be nu every time a new record is inserted and also when any part of the primary key is u then an error is raised and the transaction is aborted.

2. The referential integrity rules

The only condition for these rules is that the foreign key should either be null, or contain a value that matches an existing primary key value in the corresponding table. The rules are enforced when something is deleted from a table containing a primary key, when something is inserted into a table containing a foreign key, when the primary key is updated, or when something is deleted from a table containing a foreign key. If the rules are violated when something is deleted from a table containing a primary key then the deletion can be cascaded, making the corresponding foreign key null, or a transaction aborted. If the rules are violated when inserting into a table containing a foreign key or updating a foreign key, the foreign key can be nullified, or a new record could be added to the other table, or an error could be raised and the transaction aborted.

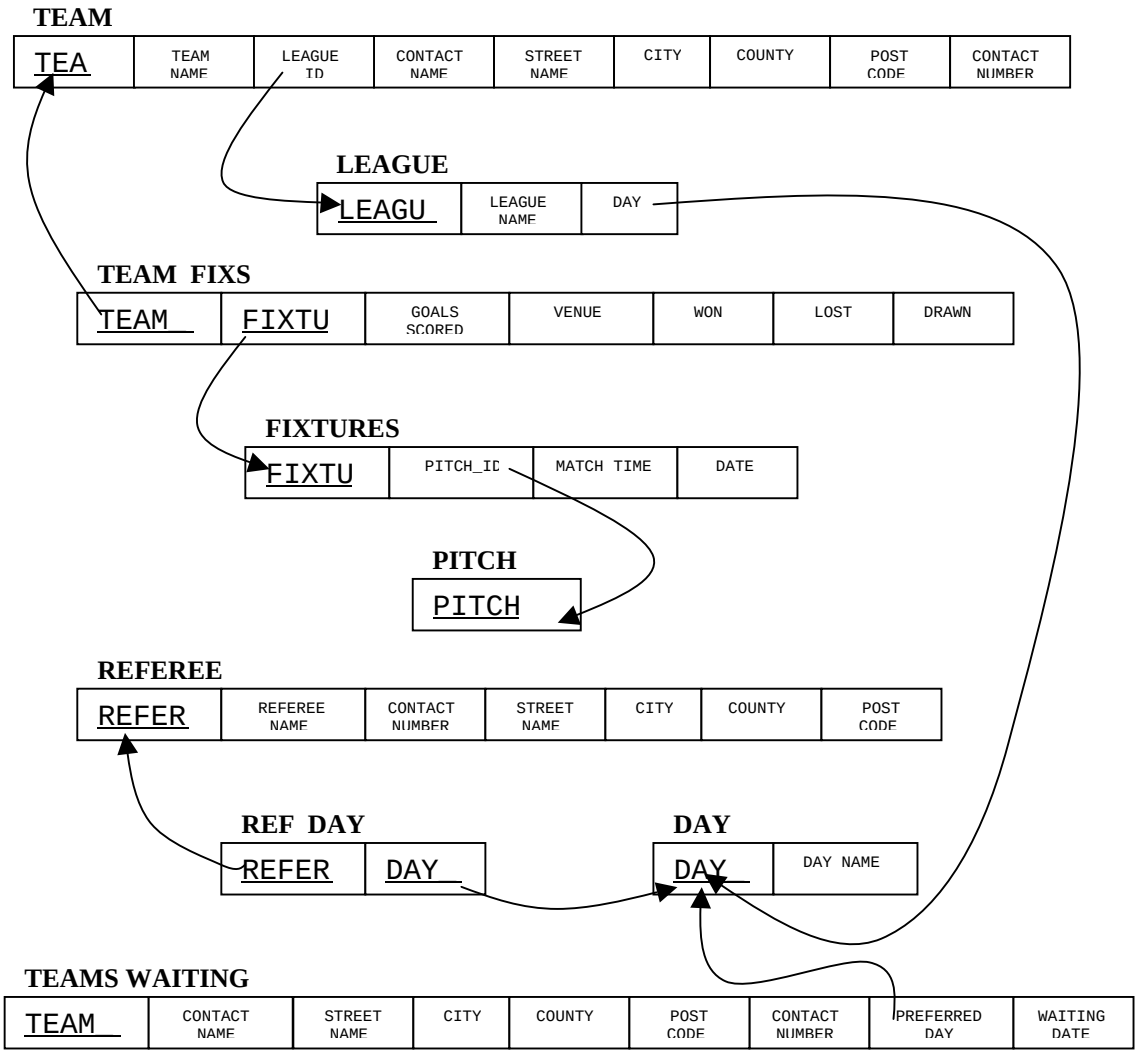
"The Relational Database Management System provides for these actions automatically, so the database designer does not have to implement the rules in the way they implement integrity checks, but to indicate which are primary and foreign keys, so that the DBMS knows where to apply referential integrity checks" ('Roberts, 1999')

The primary key for a table is created in the table design and is indicated by the key symbol next to the field name, as shown below.

	Field Name	Data Type	Description
🔑	TeamID	AutoNumber	Unique ID number for each team
	TeamName	Text	Name of team
	LeagueID	Number	Which league the team is in

Referential Integrity is enforced between all primary and foreign key pairs. An example of this is shown below where a constraint has been enforced between the LeagueID fields in the Leagues table and the LeagueID field in the Teams table. This indicates that a team's LeagueID must be one of the values that exists in the Leagues table.

Here is a diagrammatic display of the referential integrity constraints. An arc c
relation that it references represent the constraints.



3.4.4 Business Rules

A business rule is an organisational standard operating procedure for databases th
followed. Business rules ensure that the database maintains its accuracy and inte

Example business rules for the system are that no league can be running without having te
waiting list can’t be involved in a league already and old results cannot be remov
fixtures have been fulfilled.

3.5) Query Design

It is very important to ensure that the queries are produced precisely, so that they with no inaccuracies. It is essential to start by having a clear idea of what da knowing how these queries would be used later in the design.

One of the main queries needed is one that returns a list of fixtures for any giv would be to return a set of fixtures for every league and filter out the ones required forms. Once the query that produces a set of fixtures has been produced, it can that a set of results for the leagues could be returned. This idea of re-using the queries.

When designing the queries it helps to know exactly how and where the query will known that a query will be used to output data on a form, it saves time if it is shown, so then all the fields can be included in the query.

The main queries needed to retrieve all the required data are listed below:

- Show Fixtures – list of every fixture for every league
- Show Results – list of every fixture where the game has been played and a result
- League Standings – each teams record, i.e. games played, won, drawn, lost, goals
- Team/Waiting team Details – details of teams, including contact details
- Referee Details – details of referees, include contact details

Examples of how the queries were implemented are detailed in Section 4.

3.6) Form Design

The form designs are a very important part of the system, as this is what the user interact with. Human computer interaction is discussed in Section 3.7. The forms will be designed so tha satisfy the requirements of the system as identified in the user needs.

An initial 'Main Menu' form will be provided. This will open up automatically wh the user and will provide access to all other forms in the database. From the main menu, to open the following forms:

1. View Fixtures

2. View Results
3. View League Tables
4. Enter Results
5. Create New League
6. View Team Details
7. View Waiting Team Details
8. View Referee Details
9. Add Teams to Waiting List

Forms 1-3 are all fairly similar in what they allow the user to do. The user can ch
and can then decide to view the related fixtures, results or league table. The user must Once t
be able to print out the details shown on the form.

Form 4 allows the user to enter results of games that have been played. The user
which to enter the results and will be produced with the set of fixtures entered. The appro
the appropriate fixture. Closing the form will update the Team_Fixtures table. If all fixtures have
played the user should be able to renew the league so that it starts again from the or del
database.

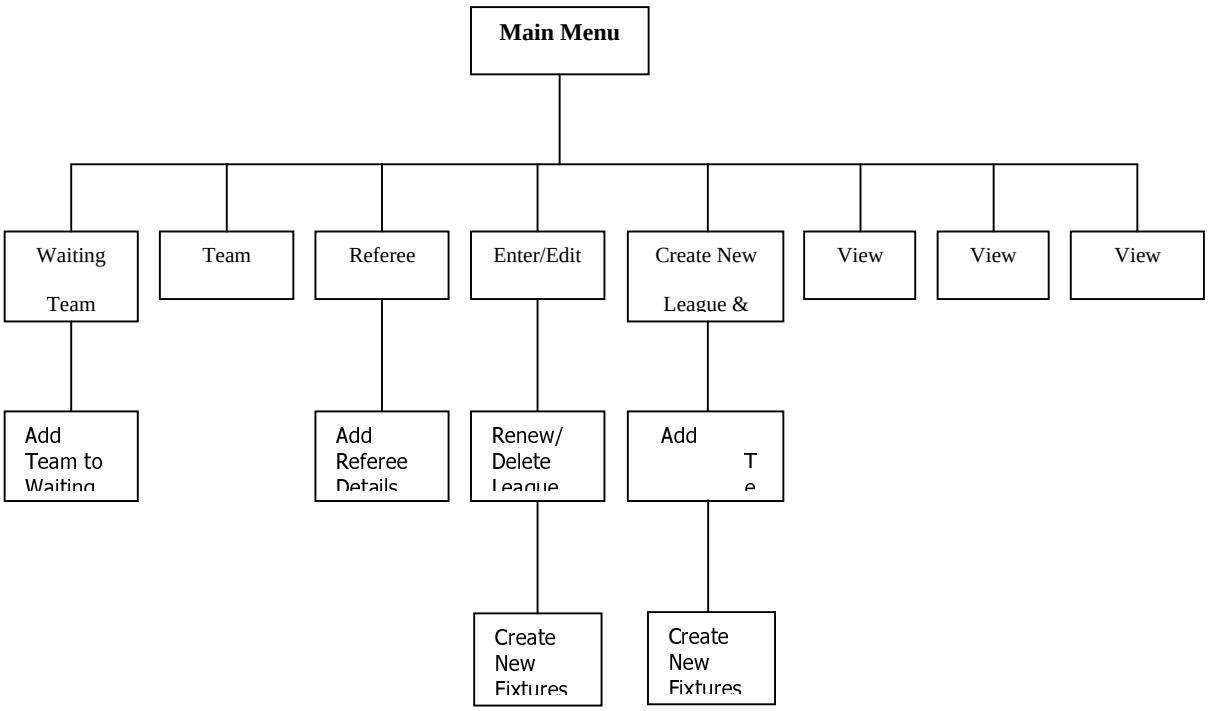
Form 5 can be used to create an entirely new league. The league is created for a
on the waiting list are entered into the league. Once the league has been created
and entered, a fixture list is created for the league.

Forms 6-8 are again fairly similar in what they provide to the user. The user can v
that are currently in a league, view details of teams on the waiting list and view details of referees
The details that are made available to the user include any contact details, for w
contact with a team representative or a referee. The form should allow the user t
specific entry can be located.

Form 9 can be used when a new team is recorded on the waiting list. The user is g
into which the details are entered and the team will be stored in the Team_Waiting table.

3.6.1 Form Hierarchy

The diagram below illustrates the flow of the forms and how they are related in the type of actions the user can make.



3.7) Human Computer Interaction

HCI (human-computer interaction) is the study of how people interact with computers and to what extent computers are developed for successful interaction with human beings. An important factor is that different users form different perceptions about their interactions and have different goals, expectations, past experiences, attitudes, motivation, personality traits, cognitive styles, and learning and keeping knowledge and skills. It is very important to pay attention to user interface design, as this holds the greatest impact on the users opinion of the system.

For the system to be a success it must have a graphical user interface (GUI). Elements of a GUI include things like menus, buttons, scroll bars and the mouse amongst others. A user's interaction with its input devices is sometimes referred to as the 'look and feel'. The systems GUI should be consistent throughout, so that the user becomes familiar with what each feature does exactly. It should be clear to the user what will happen if an action button is clicked on a form.

The user will ultimately determine how well the system has been designed. For this reason it is important to get the user involved in the design, taking any positive improvements. This will result in a more usable interface that is satisfactory to the user and is user friendly.

3.8) Web Based Front End

If this part of the system is to be implemented, the information made available of use to the user of the system. Information could be made available over the web, so that the leagues, as well as others can access it. This information could be a current league lists, which would be helpful for teams as the information would be constantly accessible use to the staff at Benyon Park as they frequently are being asked to look up details like this. Instead of having to ring a member of staff, a team could just access the web page and get the details from the web.

If, for example, the league tables were made accessible from the web, a database connection created from an ASP, and using ActiveX Data Objects and SQL query would be executed data. Then HTML and the interleaved output would be returned to the browser.

Section-4 Implementation of the System

4.1) Implementation of the Database

Implementing the database was a very time consuming process. Because the system w
requirements of Mr C Larkin and his staff at Benyon Park, he was involved during i
the system developed. As the implementation progressed each prototype of the system was
any feedback could be used to develop a more final system.

To describe exactly what was done throughout the implementation for each part of t
detailed for the purpose of this report. Instead I have focused on the main areas
examples of techniques used in these various stages.

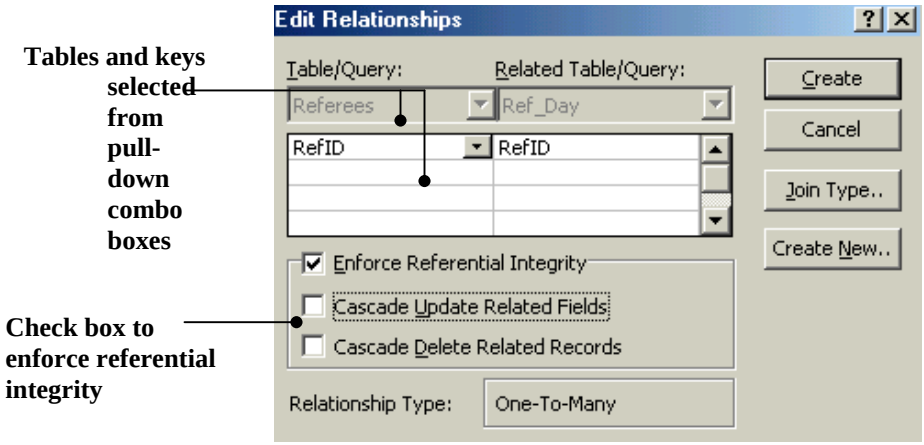
4.1.1 Implementing Tables

Using the normalised database schema produced in the design phase, implementing the tables was fa
straightforward. Each table was created using the design view in Microsoft Access
names, data types and field sizes for each field from the data dictionary defined in Section 3.2.2.

4.1.2 Enforcing Relationships

After creating the tables the next step is to enforce the relationships between an
stored in the tables has significant meaning. A relationship between two tables is created in the Rel
window. In the ‘Edit Relationships’ tool, a chosen table and its respective prima
hand columns. The related table and the respective foreign key are then entered i
The ‘Enforce Referential Integrity’ check box is selected and the relationship can the
of this is shown in the illustration below where the relationship between the ‘Referees’ ta
and the RefID field in the ‘Ref_Data’ table is created.

The relationship diagram shows all relationships that exist in the database. A co
shown in Appendix B.

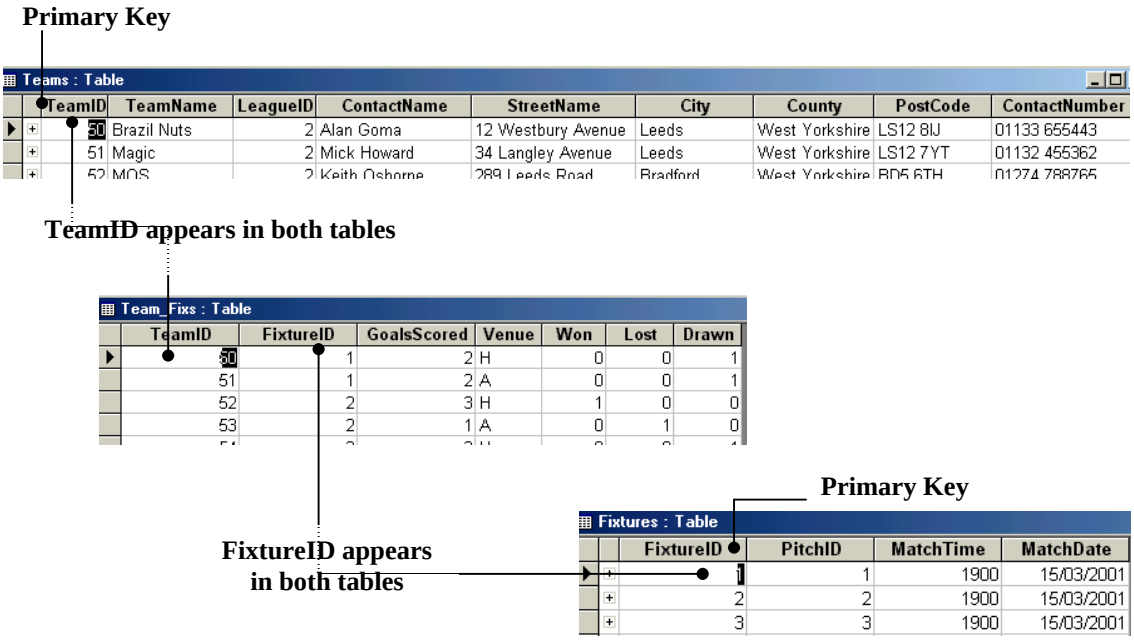


The 'Edit Relationships' tool also provides options that can be chosen which form integrity check. These are outlined below:

- Cascade Update Related Fields – When the primary key of a record is changed, any re keys will automatically be updated. This option was chosen for each relatio was not an AutoNumber, as these values cannot be changed.

- Cascade Delete Related Fields – When the primary key of a record is deleted, any rec the related foreign key will be automatically deleted. This option was chos created.

After entering data into the database, the relationships are established on the s shows how the data is related between Team_Fixtures and Fixtures tables. Any fixture involving a contains TeamID and FixtureID a record in the Team_Fixtures tables. Brazil Nuts have a TeamID of 5 is also shown to be involved in the fixture with a FixtureID of 1.



4.1.3 Implementing Database Constraints

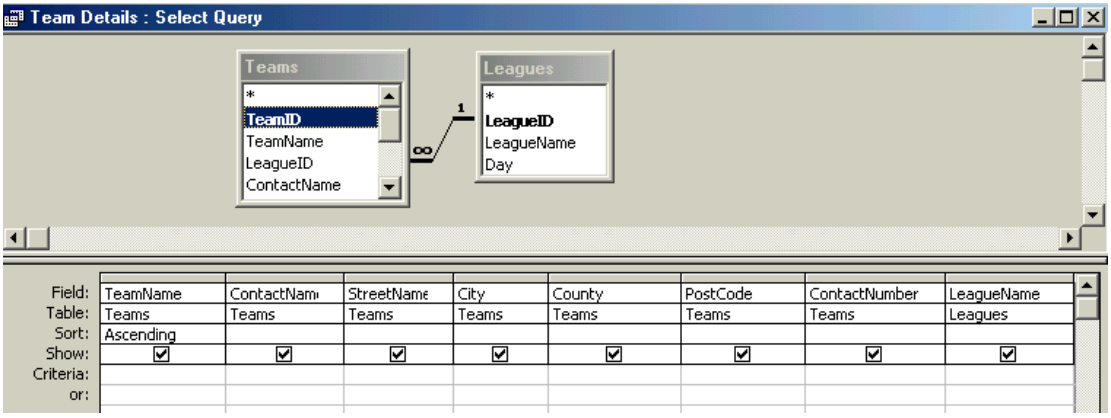
Most of the database constraints are implemented when the tables and relationships for the The integrity constraints are enforced when the field type and field size values entered for each field. The entity integrity and referential integrity rules are enforced when are created and primary and foreign keys are indicated.

Validation rules can be enforced on certain fields in the table design. For insta the Postcode fields across the tables. An input mask is a pre-defined pattern for which all data is to be a field. This input mask was created using a Wizard and the way the data could be ‘LL09 0LL’. An ‘L’ is a required capital letter, a ‘0’ is a required digit and a ‘9’ is an optional postcodes ‘BD7 6TC’ and ‘LS12 9AS’ will be accepted.

4.1.4 Query Implementation

Each of the queries was developed in a Query Design window. There were two methods required data. The first was using the design tool, where you select any table or then drag the required fields into the query builder, where different criteria, s

can be applied to the fields. Below is the query that is used to find all of the te
Teams and Leagues tables. It also shows that the records retrieved from executing the
alphabetically by TeamName.



The second approach used was producing the queries using SQL. There were many di
commands used to design all of the queries, including SELECT, DELETE and UPDATE.
designed that would update the Team_Fixs table for new results get entered into the database.
Lost and Drawn fields, are used to indicate which team won the fixture, and is al
used by a query that creates the league tables, where the values are summed to get
needs a '1' inserting into the Won field and a '0' into the Lost and Drawn fields. The losing team needs a
inserting into the Lost field and a '0' into the Won and Drawn fields. If the match ends in a draw, each
'1' inserted into the Drawn field and a '0' in the other two fields.

To solve this problem three queries were created, one to update the Won column, another to update the Lost
column and third to update the Drawn column. The three queries are almost identical, with only the obvious
differences. Below is the SQL query for updating the Won column:

```
UPDATE Team_Fixs, Team_Fixs AS Team_Fixs_1
    SET Team_Fixs.Won = 1
        WHERE ((([Team_Fixs].[FixtureID])=[Team_Fixs_1].[FixtureID]) And
            (([Team_Fixs].[GoalsScored])>[Team_Fixs_1].[GoalsScored]) And
            [Team_Fixs].[TeamID]<>[Team_Fixs_1].[TeamID]);
```

The query locates the two unique teams that shared the same fixture and sets the value in the Won field to '1',
for the team that has a greater GoalsScored value. Here are examples of the queries that were cre
described later in this chapter.

4.1.5 Fixture Generation

Part of the user requirements was to develop the feature of being able to create a
teams. This part of the system had to be created early on in the project, as most other fea
fixtures for a league. My original thoughts were to create an algorithm that wou
of teams that were entered into a league. This however proved to be much harder than expected; as it
that a rather complex spanning tree algorithm would have to be implemented. After
Mr Larkin it became evident that the leagues running at Benyon Park contain either
reason for this is to optimise income throughout the year. A league with six teams
twice would span ten weeks. This would mean that the league could be renewed five
weeks over holidays or to run tournaments. A league with eight teams, playing e
twice would last for fourteen weeks, meaning it can be run three times over a year
group competitions. The centre also wishes to avoid having an odd number of teams, as one team
out every week.

This information made the fixture generation far simpler to implement. The implem
through using an event procedure assigned to a command button on the form (the reasons will be
explained later). Event procedures were created using Microsoft Visual Basic provided b
fixtures, the number of teams to be entered, the start date and the match time nec
largest FixtureID currently in the table. This information was taken from the values pro
query used to create the form. Depending on the number of teams entered the proce
for six or eight team. Below is the code that shows how the first week of fixtures is crea

```
'Create first round of fixtures
DoCmd.RunSQL ("INSERT INTO Fixtures(FixtureID, PitchID, MatchTime, MatchDate)
VALUES('" & fixNum & "', '1', '" & startTime & "', '" & matchDate & "');")
DoCmd.GoToRecord acDataForm, "CreateFixtures", acFirst
InsertHomeTeam (fixNum)
DoCmd.GoToRecord acDataForm, "CreateFixtures ", acGoTo, 2
InsertAwayTeam (fixNum)

fixNum = fixNum + 1
DoCmd.RunSQL ("INSERT INTO Fixtures(FixtureID, PitchID, MatchTime , MatchDate)
VALUES('" & fixNum & "', '2', '" & startTime & "', '" & matchDate & "');")

DoCmd.GoToRecord acDataForm, "CreateFixtures", acGoTo, 3
InsertHomeTeam (fixNum)
DoCmd.GoToRecord acDataForm, "CreateFixtures", acGoTo, 4
InsertAwayTeam (fixNum)
```

```
fixNum = fixNum + 1
DoCmd.RunSQL ("INSERT INTO Fixtures(FixtureID, PitchID, MatchTime, MatchDate)
VALUES('" & fixNum & "', '3', '" & startTime & "', '" & matchDate & "');")
DoCmd.GoToRecord acDataForm, "CreateFixtures", acGoTo, 5
InsertHomeTeam (fixNum)
DoCmd.GoToRecord acDataForm, "CreateFixtures", acGoTo, 6
InsertAwayTeam (fixNum)
```

Thus for each fixture an SQL command is run to create a record for a new fixture. The fixture is then located and the home and away teams are created for the fixture. The InsertHomeTeam and InsertAwayTeam functions, enter the home and away teams into the Team_Fixs table respectively.

```
Function InsertHomeTeam(fixNum)
    DoCmd.RunSQL ("INSERT INTO Team_Fixs(TeamID, FixtureID, Venue)
    VALUES(forms!CreateFixtures.TeamID, '" & fixNum & "', 'H');")
End Function
```

After the fixture details have been entered for each week, the AddWeek function is run. This calls a function is used to increment the match date for every week. This is a complex function as it months, years and also checks for leap years. The entire function is included in Appendix C.

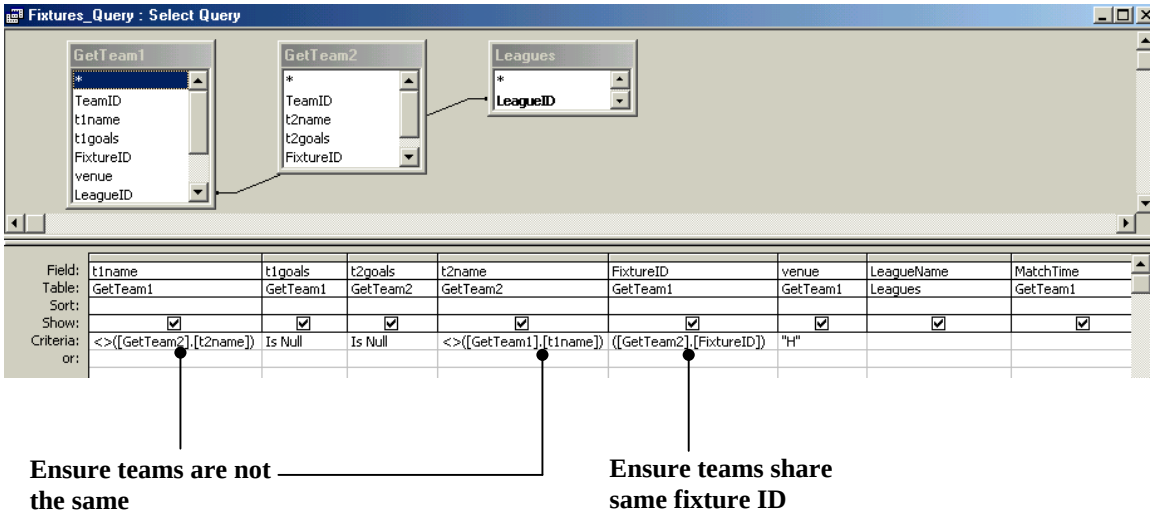
The process of creating fixtures using this method makes the job extremely simple. The user will enter teams have been chosen for the league the user will enter a start date for the matches and click the button that will automatically do the work for them. A complete fixture list can be created at each match is allocated a time, date, and pitch number without the user having to do anything.

4.1.6 Managing Fixtures/Results

Once the fixture details were available from the tables, a way was needed to manipulate them so that a fixture could be viewed in the proper manner (i.e. Team 1 vs. Team 2). This was necessary so that the result could be displayed on a form for the user and also so that the user could print a fixture list.

This problem was solved using a more complex query than previously used. Initially the query returned the team and fixture details for each individual fixture. Not surprisingly this was not empty. A second query was then created that contained the same fields as the first. The two queries and their fields were then made available. The layout of a fixture was then created in a table applying criteria to the fields to ensure that each fixture returned was correct.

When the problem of manipulating the fixtures into a desired format was solved, th
solve the problem of handling results of the fixtures. The diagram below shows ho
created. To return results instead of fixtures, the criteria for t1goals and t2goals simply changed from ‘Is Nu
to ‘Is Not Null’.



4.1.7 Maintaining League Tables

Another major feature of the system was to maintain league tables. This would give the
ability to view and print league tables, which provides important information to c
league table needs to contain the following details for each team in the league; t
played, games won, games drawn, games lost, goals for, goals against, goal differe
gained.

This problem was solved using a selection of queries. An initial query was designed
details as well as performing sums of the fields for each team. The Team_Fixtures table contained
and Drawn fields provide a total of games each team won, lost and drawn. The sum of e
GoalsScored column provides the ‘goals for’ for each team and a count of the same fie
used to indicate how many games each team has played. A similar query for each team
new query was created that summed all of the entries of goals scored by the opposi
fixtures. The ‘goal difference’ was a simple subtraction of ‘goals against’ from
projected assigning three points for a win and one point for a draw in the calcula

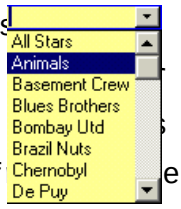
The league table is then sorted in order of points. If the points are equal, rec
highest goal difference and then a second sort is applied so the third order of precedence
highest number of goals scored.

4.1.8 Implementation of Forms

Implementing the forms was the trickiest part of the database creation. The forms
queries and their records as the control source. In each of the forms, different
Basic (VB) commands were created to manipulate the data.

The first three forms were all very similar in the way they were created. The forms
details, view waiting team details and view referee details. The purpose of the
from the relevant queries set out in a manner that is easy to view for the user.
form was opened, was to list all records. These forms are of benefit to the user
may need to find a phone number for a particular team in a certain league, or the
waiting list.

A feature that was included in these forms, gave the user the ability to perform a sear
record, i.e. finding a record using team name, contact name or league name fields
set of select boxes, where for example the Team Name drop down box (shown here)
every team name from the records on the form to choose from. entry is
the form will return only the effected record(s) to the user. s implemente
function written in VB. This function finds the value to search for, f elect box
the form. In the example code below the filter value is set to the team name tha
the user. The filter is then performed with the command Me.Filter = sFilter and the filtered values are
returned to the user.




```
Sub DoFilter()
Dim sFilter As String
sFilter = ""
If Not IsNull(FilterByTeamName) Then
    sFilter = "TeamName=" & "'" & (FilterByTeamName) & "'"
End If
Me.Filter = sFilter
Me.FilterOn = True
End Sub
```

The next three forms were also very similar in the way they were created. These w
fixtures, view results and view league tables. Before any of these forms can be v
performed on the relevant queries so that records for only the chosen league are returned.
options are chosen, a form is opened with the list of leagues that are currently runni
command button placed next to each league, which when selected, the form was opened with
records relating only to the chosen league. The link is made using the code belo
perform the filter is taken from the
LeagueID of the chosen leag

```
stLinkCriteria = "LeagueID=" & Me![LeagueID]
DoCmd.OpenForm "Show Fixtures", acNormal, acEdit, stLinkCriteria
```

Each form layout was formatted so that the output was made viewable in the best wa
the details. The user is then given the option to preview the results or print the league table p
which can be used to distribute around the centre on notice boards or to the teams
played. The details are printed using the following example code, which shows how
is executed when the print button is selected.

```
Dim stLinkCriteria As String
stLinkCriteria = "LeagueID=" & Me![LeagueID]

Dim stDocName As String
stDocName = "ViewFixturesReport"
DoCmd.OpenReport stDocName, acNormal, , stLinkCriteria
```

The link criterion is again selected using the LeagueID which is then applied as a filter on the repo
printing the fixtures. The last line of the code opens and prints the report auto
preview.

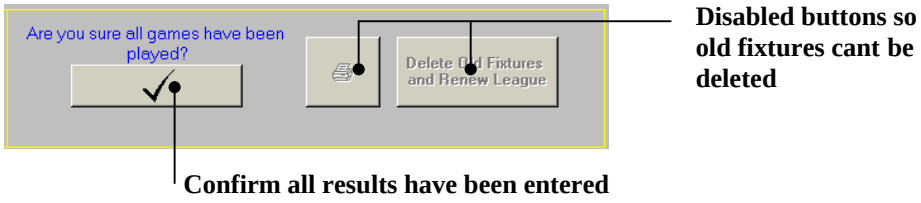
On the ‘View Results’ form the fields values for the goals scored by each team are not ena
so that the user can’t accidentally alter the results.

A form was also created so that the user could enter and edit results. This form is presented before
user where a league is chosen for the results to be entered/edited. When opened
of all fixtures and results for the chosen league which have been filtered from th
In this form the fields for each teams goals is made enabled so that data entry is made pos
appropriate results have been entered, the ‘Exit and Update Scores’ button will c
queries described in section 4.1.4 update the ‘Won’, ‘Drawn’ and ‘Lost’ columns in the Team_Tests table.

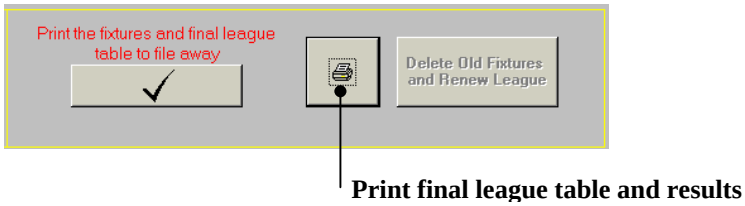
If all games have been played and each fixture has a corresponding result, then th
renew or delete the league and fixtures, so that a new league can be started up.

Prompts are provided to the user to make sure that it is the desired action to renew
Carrying out either of these options means that all previous fixtures and results
to remove unwanted data. It is therefore necessary to make sure that the user is positive that th
proceed. The way this was done is shown in the diagram below:

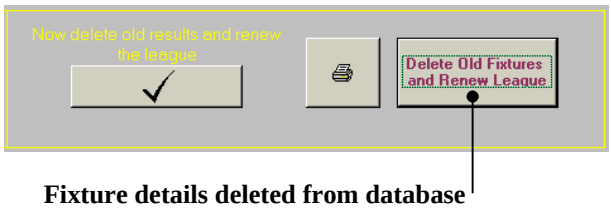
Step 1


Are you sure all games have been played?
Delete Old Fixtures and Renew League
Disabled buttons so old fixtures cant be deleted
Confirm all results have been entered

Step 2


Print the fixtures and final league table to file away
Delete Old Fixtures and Renew League
Print final league table and results

Step 3


Now delete old results and renew the league
Delete Old Fixtures and Renew League
Fixture details deleted from database

The user clicks the ‘confirm’ button to be able to renew or delete the league, which
enabled. A button is made enabled using the following VB command, which shows ‘pr

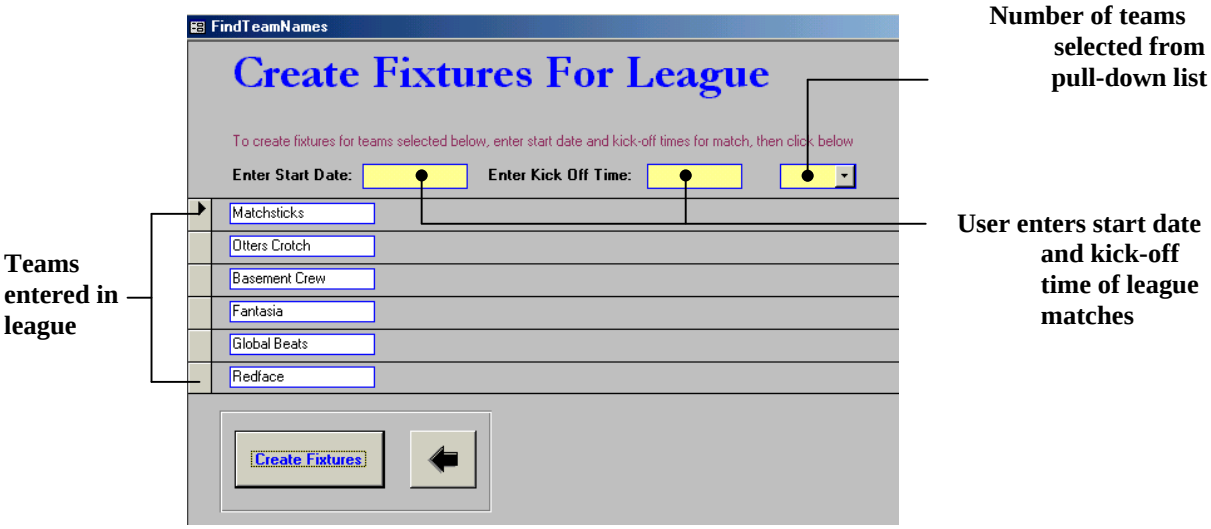
```
printResTab.Enabled = True
```

Before the user can proceed and renew or delete the league, a copy of the final le
printed with the ‘print’ button. These will be used for storage so that if any qu
about the league, they can be referred to. The ‘delete’ button, when selected, de
the database by removing records from the Team_Fixtures and Fixtures tables. This is carried out using t
below. A ‘while’ loop is used to go through each fixture on the form and SQL comm
tables with the FixtureID

```
Dim n As Integer

        n = 1
        While (n < (FixCount + 1))
            DoCmd.GoToRecord acDataForm, "ShowFixtures_Results", acGoTo, n
            DoCmd.RunSQL ("Delete * From Team_Fixs Where FixtureID =
            Forms!ShowFixtures_Results.FixtureID;")
            DoCmd.RunSQL ("Delete * From Fixtures Where FixtureID =
            Forms!ShowFixtures_Results.FixtureID;")
            n = n + 1
        Wend
```

The user is then given two choices of what to do with the league, it can either be renewed or the league can be removed from the league list. If they have decided to renew the league then a team can be removed from the league if they have decided a new team can be added from the waiting list, where the teams for the day that the chosen next (explained later in this section). If the league is not to be renewed, the user chooses to remove team and league details are completely removed from the database.

When the league is renewed, the user first selects the teams that are to be renewed for the teams, or teams can be added or deleted, depending on how many of the teams again. When the 'Create Fixtures' button is selected the teams chosen are shown to the user with text boxes where they are expected to enter values for the start time and start date, and kick-off fixtures are to be created for.


FindTeamNames
Create Fixtures For League
To create fixtures for teams selected below, enter start date and kick-off times for match, then click below
Enter Start Date:
Enter Kick Off Time:
Matchsticks
Otters Crotch
Basement Crew
Fantasia
Global Beats
Redface
Create Fixtures
Number of teams selected from pull-down list
User enters start date and kick-off time of league matches
Teams entered in league

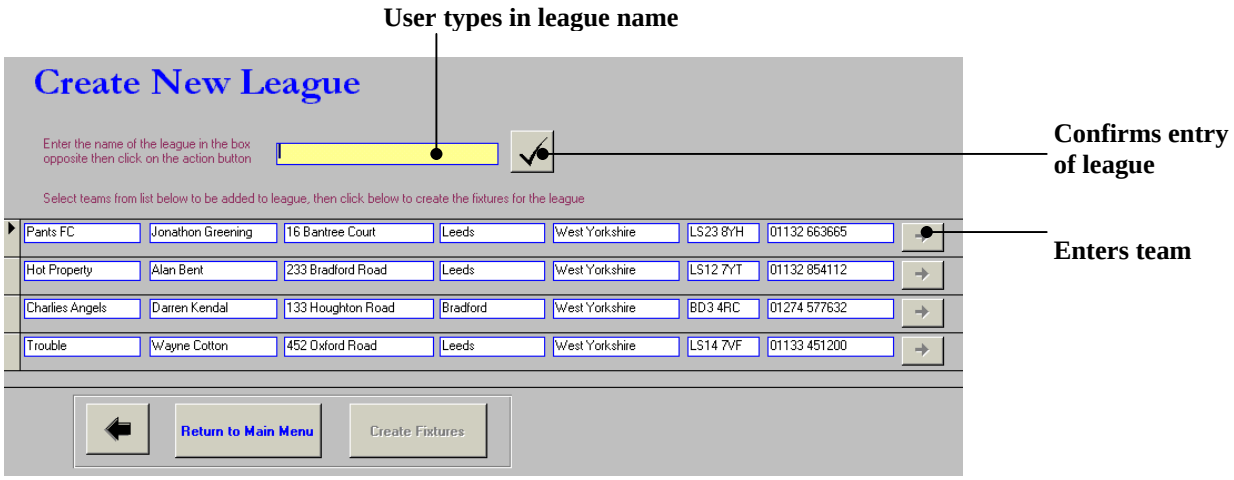
The 'Create Fixtures' command button executes the function detailed in section 4.5.5 creates a whole fixture list for the league programme. The benefits of these forms give the user nights fixtures. When the league programme has finished the user can then renew or consummate ease. It's essential that the user can perform these operations, to re can play the week after the final league. This maximises income in the centre.

Another form that is required by the user, is one that can create a brand new leag
list. The user chooses to create a new league and is given the option of which d
through a form containing the list of days. When the day is chosen a form is oper
teams on the waiting list that are wanting to play on that particular day. Applyi
day as criteria provides these records.

To create fixtures for a league, the league must first of all be created and then
league. As certain buttons are disabled, the only option for the user is to type the name into the first
provided text box.

```
Dim i As Integer
i = Forms!CreateNewFixs.MaxLeague
i = i + 1
DoCmd.RunSQL ("INSERT INTO Leagues(LeagueID, LeagueName, Day) VALUES(" & i & ",
forms!CreateNewFixs.NameOfLeague,forms!CreateNewFixs.DayID);")
MsgBox "You have created a new league - ID: " & i & ""
```

Clicking the 'action' button inserts the League details into the table using the code above. The max
LeagueID is found and the new league's ID number is assigned the next number. The
user is to enter the teams into the league, i.e. they can't exit the form or enter
are disabled. The user enters each team with the relevant action button, which en
Teams table with the LeagueID value set to the one just created.



Once the teams have been entered into the league, the fixtures are created using
earlier in this section. These forms benefit the user, by checking through the waiting list and the
is a free slot in the timetable a new league can be created in seconds.

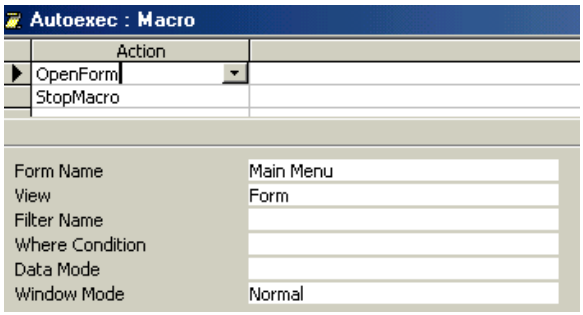
The final form created for the user is used to enter a team onto the waiting list record from the relevant query. Navigation buttons can be used to browse through th command button can also be used to delete a record, which is important for when a longer commit to playing. The last command button that can be used creates a new the details of the team. It was decided, on instruction from Mr C Larkin that tea include a date field. This is used to enter the date when the team joined the wai the team who joined the list first is entered into a new league first, operating a

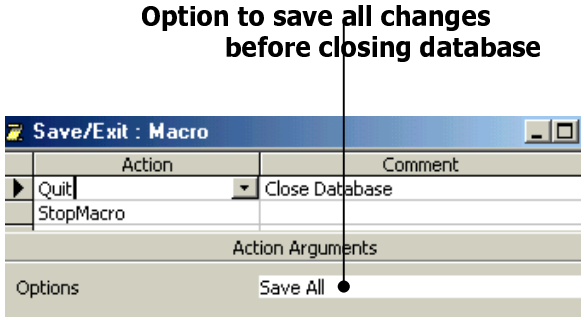
4.1.9 Implementation of Reports

The reports were created to design the various printouts needed by the user. The similar to the way the forms were created. The Wizards provided by Access were us values to be used on the reports were taken from the relevant queries. The repor where the prints are made. Example reports are shown in Appendix D.

4.1.10 Macros

Two macros were produced to aid with the usability of the system. It was importa opened automatically without user intervention. This provides the user with a friendly hides away low-level design features like tables and queries. The problem was solved *Autoexec* (shown below). The macro opens the chosen 'Main Menu' in form view and th recognises 'Autoexec' and runs the macro as soon as the database is opened.





The second macro created provides the user with the ability to shut the database a button was created to exit the system. The command button was designed to execute event procedure, which in turn executed the Save/Exit macro.

4.2) Implementation of Web Based Front End

Once the database was completely implemented it was then decided that it would be page as it would make the different league tables available for the customers to view

Active Server Pages, saved to an area on the CSSQL1 Server on the university compu
as follows:

1. *Datastore.asp* provided connection to the database
2. *Index.asp* creates a list of the league names, which are links to open the lea
3. *Get_table.asp* outputs the chosen league table

These files are included in Appendix E.

A connection was made to the skylark database using the command below:

```
strConnect="Driver={Microsoft Access Driver(*.mdb)};
   DBQ=g:\Inetpub\wwwroot\cszajs\Test\skylark.mdb"
```

The *index.asp* file was used to find all the league names currently running at Benyon P
for the user to click on to open the corresponding league table. Firstly an object
made to open the Leagues table from the connected database:

```
Dim objRS
Set objRS = Server.CreateObject("ADODB.Recordset")
objRS.Open "Leagues", strConnect, adOpenDynamic, adLockReadOnly, adCmdTable
```

A command was then created to go through each record in the record set outputting
links. This is the command used:

```
While Not objRS.EOF
Response.Write "<a href = get_tables.asp?name=" & objRS("LeagueName") &_
        ">" & objRS("LeagueName") & "</a><br>"
objRS.MoveNext
Wend
```

The get_table.asp file was used to output the league table for the chosen league. The l
index by the user is saved as a query. A query was then created to find the information neede
table (this query was similar to the one in the database), where the league name
the required records.

```
strName = Request.QueryString("name")
strQuery = "SELECT TeamName, Played, Won, Drawn, Lost, Points, LeagueName
        FROM webTables WHERE LeagueName = '" & strName & "'
        ORDER BY Points DESC"
```

A record set was then opened containing the records found by the query:

```
Set objRS = Server.CreateObject("ADODB.Recordset")
objRS.Open strQuery, objConnection
```

The next command then goes through each of the records in the record set outputti
the league table. Each record is output with a border to create the effect of a

```
While Not objRS.EOF
    Response.Write "<TR>"
    Response.Write "<TD>" & objRS("TeamName") & "</TD>" & "<TD>" & objRS("Played")
        & "</TD>" & "<TD>" & objRS("Won") &"</TD>" & "<TD>" & objRS("Drawn")
        &"</TD>" & "<TD>" & objRS("Lost") &"</TD>"
        & "<TD BGCOLOR = 'yellow' >" & objRS("Points") &"</TD>"
    Response.Write "</TR>"
    objRS.MoveNext
Wend
```

The league tables can be viewed using the URL http://www.leeds.ac.uk/Screen/test/ and screen shots of the Web
page in action are shown in Appendix F.

Section-5 System Testing

This section documents the process of testing the final system. Testing is requi
system works and is used at key checkpoints in the overall process to determine wh
been met. The system must be subjected to performance tests, using benchmarks tha
combination of work that attempts to imitate the kinds of thing the system will do

Two types of testing will be carried out on the system, white box testing and black box testing.

- White Box Testing is concerned with testing the software product and will disco
commission, indicating any parts of the implementation that are faulty. It
complete specification has been implemented.

- Black Box Testing is concerned with testing the specification and will discover
indicating any parts of the specification that have not been fulfilled. It
of the implementation have been tested.

In order to fully test the system both black and white box testing are required.

5.1) Black Box Testing

Black Box testing was carried out on the system where data entry was required. A
range of expected, erroneous and extreme data were entered into the fields to ensu
it would be expected to. Where defects were found, alterations were made where ap
testing that was carried out:

Enter Results				
Goals Scored	7	Accept	As expected	
	Red Devils	Reject – produce error message	As expected	
	27/6/01	Reject – produce error message	As expected	
Create Fixtures				
Start Date	27/6/01	Accept	As expected	
	2130	Reject – produce error message	As expected	
	35/1/01	Reject – produce error message	As expected	
	Red Devils	Reject – produce error message	As expected	
	<Null>	Reject – produce error message	As expected	

Kick Off Time	2130	Accept	As expected	
	27/6/01	Reject – produce error message	As expected	
	Red Devils	Reject – produce error message	As expected	
	<Null>	Reject – produce error message	As expected	
	7:45	Reject – produce error message	As expected	
	21300	Reject – produce error message	Accepted erroneous match time	Input mask created on text box only in format ‘0000’
	7896	Accept	As expected	*
Select Number of Teams	6	Accept	As expected	
	17	Reject – produce error message	As expected	
	Red Devils	Reject – produce error message	As expected	
Create New League				
League Name	Monday Night	Accept	As expected	
	27/6/01	Accept	As expected	*
	17	Accept	As expected	*
	<Null>	Reject – produce error message	As expected	
Add New Team				
Team Name	Red Devils	Accept	As expected	
	27/6/01	Accept	As expected	*
	31	Accept	As expected	*
Contact Name	Andy Smith	Accept	As expected	*
Contact Number	01274 783551	Accept	As expected	*
Street Name	18 South Drive	Accept	As expected	*
City	Leeds	Accept	As expected	*
County	West Yorkshire	Accept	As expected	*
Postcode	LS12 4RD	Accept	As expected	
	Andy Smith	Reject – produce error message	Accepted invalid input	Input mask created on text box only in format ‘LL09 0LL’
	01274 783551	Reject – produce error message	Accepted invalid (As above)	
Date	27/6/01	Accept	As expected	Input mask created on text box only in format ‘09/09/09’
	35/1/01	Reject – produce error message	As expected	
	Andy Smith	Reject – produce error message	As expected	
Preferred Day	1	Accept	As expected	
	8	Reject – produce error message	As expected	
	Red	Reject – produce error message	As expected	

* Some fields allow any text to be entered. It’s up to user to make sure correct data is

5.2) White Box Testing

White Box testing was carried out by performing certain tasks that a user would ex
created system. Various acceptance tests were carried out, and cross referenced t
that all aspects of systems have been implemented.

- Can the system add a team to the waiting list?

I chose to ‘Add Team to Waiting List’ from the main menu and then clicked on the ‘Add Recor
provided me with a new empty record to add the teams’ details to. The details wer
and then processed. This helps satisfy User Needs 1.7 and 1.3.

- Can the system create a new league with fixtures for teams?

I chose to ‘Create New League’ from the main menu and chose to create the league t
a name for the league with the ‘confirm’ button, which then made it possible to en
teams were entered into the league and then the ‘Create Fixtures’ button was click
league and the kick-off time of the games were then entered in the appropriate fields and
number of weeks. The ‘Create Fixtures’ was then pressed, which automatically created
were then viewed to make sure everything was processed correctly. This helps sati

- Can results of a fixture be entered and updated in the system and the league table?

I chose to ‘Enter New/Edit Results’ on the main menu and chose to enter results fo
League’. This presented me with the set of fixtures and results for the league. I
taking place on the ‘12/04/01’ and then exited the form. I then chose to view the le
it was apparent that the league table was updated. This helps satisfy User Needs 1.9, 1.10 and

- Can the system print out details of fixtures, results and tables?

From the main menu, in turn, I chose to ‘view fixtures’, ‘view results’ and then
‘Monday Masters League’. Whilst on the forms I selected the print button, which p
information. This helps satisfy User Needs 1.4, 1.5 and 1.6.

- Can the system view contact details of teams and referees?

From the main menu, in turn, I chose to ‘view team contacts’, ‘view waiting team c
contacts’. Whilst on these forms I performed searches for particular teams, which brought up
details. This helps satisfy User Needs 1.1, 1.2 and 1.3.

- Can the system renew a league?

Here I entered in all results for the ‘Wednesday Corporate League’ and confirmed that all had been entered. I then followed the instructions by printing the results and league tables and fixtures. From here I chose to delete one of the existing teams and replaced it with another. I then confirmed the start date and match time and created the fixtures as before. This helps satisfy the rest of User Need 1.12.

- Can the system delete a league?

Again I entered all of the results for the renewed ‘Wednesday Corporate League’ and confirmed that all had been entered. I then followed the instructions by printing the results and league tables and deleted the old fixtures. I then chose the option to delete the league. When I was asked which league it wasn’t to choose from. This helps satisfy the rest of User Need 1.12.

5.3) Conclusions of Testing

No major problems occurred whilst doing the testing. Apart from a few basic changes to the wording on the forms, nothing had to be altered in the code. This was very important, as the consequences of test failure could have proven to be costly. A failure of a white box test may have resulted in a redesign and black box tests to be repeated and white box paths to be determined. This ensures that the design and production stages were carried out carefully and concisely. Because of this the testing was relied upon to discover any software, instead it was used to confirm that there were very few faults present. The test of quality assurance rather than quality control.

Section 6 – Demonstration and Feedback

6.1) Details of Demonstration

It was decided that the user, Mr C Larkin, required a demonstration of the system to
his satisfaction and that each of the features could be operated with ease. A dem
more useful than an extensive user manual, as there are only a small number of pos
someone who has used the system before can then train any user that is new to the sy

A copy of the system was taken to Benyon Park where the demonstration was held. A
demonstration was made for each feature of the system, explaining the purpose of
button and what order different processes are carried out in.

Mr C Larkin was then given the opportunity to test out the system for the first ti
used the system was watched very carefully, its actions made were unfamiliar to how
them to be. I encouraged him to speak any thoughts, which was useful since interf
obvious to the passive user.

6.2) Feedback from Demonstration

The feedback from Mr C Larkin about the system was very encouraging and it was evident
with the outcome. His initial comments were made about the visual side of the pro
well laid out and was pleasing on the eye with no harsh, annoying colours. The consistent layout of
made them easy to interpret. Also praised was the ease of use of the product. It was noted that it w
to him what each button on the forms were for and it was easy to understand how to carry out tasks, like
creating a new league, were performed.

Mr Larkin thought that the product's potential to accomplish the goals set by him,
accomplished, and extra features like searching for a particular team's details wo
Another positive comment made was about the apparent time that the new product wou
performing certain tasks.

Section 7 – Evaluation and Future Developments

This section examines the overall system, identifying any limitations and providing a description o enhancements that could be made.

7.1) Evaluation of the System

The system created provides for each of the User Needs that were clearly set at th well as incorporating new features that improved the functionality and usability of the syste

The user can easily view the contact details of the teams and referees, which was facility was then provided so that the user was able to search for a particular entry, using relevant fields, for instance team name. This proved to be a very useful extra fe end user was very pleased with. However, once the user has found the entry needed information directly from the screen. This could create problems as copying the i transcription errors. It would probably be beneficial if the details could be pri

Both the user and myself feel the system provides a good method for viewing fixtures, resu league tables. These details can be accessed with ease and they provide a well laid out in a comprehensible manner, giving the user all the information they need to know, for example pitch number of a game. These details are updated automatically when new results system and printouts can be taken within seconds of a game being finished, so copi customers. It would be hard to improve these aspects of the system.

The system provides the user with a simple and easy way to enter and edit results any given night at Benyon Park will be in charge of entering the scores of the games. The Manager. This means there are no problems with teams having incorrect scores ente teams to benefit. If there are discrepancies with results that have been entered, both teams as well as the referee's score sheet to find the correct score. At the mo results to be entered for each league separately and the user is presented with ev the league, where the user can find the fixtures using the date for the game. As more th be run on the same night, it could be beneficial to have a list of all games scheduled particular night, i.e. the night the results need to be entered. This would only be presented games of significance when it comes to entering results at the end of the night. prevent mistakes being made when entering results, because at the moment the user entering goals scored by only one team. Even though the mistake will eventually b team will have played one game less than the others in the league table, it would to make sure both scores have been entered for a result. It is impossible to ensure t result have been entered correctly and is therefore the users responsibility to ma

method provided for entering satisfactory, but there are a few possible alterations the current system.

The system provides the user with a simple step by step process for creating a new league, which p user from making a mistake. The new league can be set up within minutes, with a co fixtures. The only limitation for this part of the system is that fixtures can on Although this satisfies the User Needs for now, in the future it will be required to allow more teams t entered into a league. To solve this problem the best approach would be to create creates a fixture list for any number of teams. This caused me several problems decided it would be too difficult to solve in the time remaining, with all other a concentrate on.

Another User Need was that new teams could be added to the waiting list. The syst do this, as the entry can be done instantly straight onto a form, which is then en Currently a team can only be added to the waiting list for one particular night. play on a number of different nights and will take the first available slot, so providing for customers in the long run. The customers could also indicate whether they wish to times, which has a factor on how much they would pay to play.

7.2 Future Enhancements

Many improvements to the system have been considered. Section 7.1 could be later implemented future versions. Below are further examples of improvements that could help impro

- Create Cup Competitions

Giving the user the ability to create competitions to run in between the league progr would allow the user to enter a number of teams and automatically produce a set of format. This would take account of the variety of leagues involved in different leagues.

- Provide Statistics on Team Performances

This feature would track the results for each team in a league to give an indicati performing over a certain number of weeks. The database could track the last five results of a team a new table for the teams, based on their form. This information could then be gi how well other teams have been doing over recent weeks.

- Create Templates for letters to be sent from Information in the System

Information in the database could be used to automatically create letters that hav
be a renewal form where the team is written to, to ask whether or not they wish to
league. The team contact details could be retrieved from the system and sent strai
example would be to track a team that hasn't shown up on the night to play. Again
sent to an invoice letter to the offending team.

- League Pricing

Some of the leagues running at Benyon Park have different pricing plans. This is
run off peak and others run at peak times. Leagues entered during peak times are na
Another field could be entered into the database to record how much it would cost
This would help track how much the leagues are priced at, i.e. how much they shou
would be calculated using the number of teams entered and will help give an idea of which le
best business.

- Use in Different Areas

Due to the consistent and generic qualities implemented in the system, it would be
different sports centres to manage different sports like tennis and rugby. During the
opened a new centre in Huddersfield where this system could easily be implemented

7.3) Conclusions

I feel that this project has, overall been a success. All initial user needs hav
adhered to and also further enhancements were made to improve the systems' overall
though the final system does meet the requirements it is still important that bugs can be found a
eradicated in further versions to make it even better.

It is hoped this system is deemed a success by the management team at Benyon Park
it in the future to help them manage the leagues with greater efficiency and in a more cost effective way.

References

Deitel & Deitel (1998), C++ How to Program. In: Operator Overloading, pp.502-502, Prentice Hall.

Elmasri & Navathe (2000), Fundamentals of Database Systems, Addison.

Garcia-Molina, H & Ullman, J.D (1999), Database System Implementation.

Hobley, K (1998), Introduction to Human Computer Interaction, School of Computer Studies, University of Leeds.

Jesty, Eur Ing Peter (1999), Software Project Management, School of Computer Studies, University of Leeds.

Mott, Peter & Roberts, Stuart (1998), Introduction to Databases, School of Computer Studies, University of Leeds.

Mott, Peter & Roberts, Stuart (2000), Advanced Databases, School of Computer Studies, University of Leeds.

Mott, Peter & Roberts, Stuart (2001), Distributed Information Management Systems, School of Computer Studies, University of Leeds.

Roberts, Stuart (1999), Database Principles and Practice, School of Computer Studies, University of Leeds.

Appendix A Project Experience

I have found undertaking this project to be both an enjoyable and valuable experie
skills including time management and have also learnt new technical skills that w
come.

If I were to give any advice to somebody starting a project it would be to get an ear
underestimate the amount of work that is in involved in managing a whole project.
semester, with a particular piece of coursework, as much time was spent on my project as I had h
As a consequence my progress through the project suffered and a great deal of time
which had a knock on effect on other modules and the work began to pile up.

I feel that it is also important to choose an area for the project that interests you.
has to be spent doing the project and spending the time on a subject of interest w
was very rewarding to see the final system that was created.

It is also vital to use the help that is offered to you by your project supervisor
are headed in the right direction, which makes sure the time you spend on the proj
a project where a final result is unknown, I would urge you to form a good working relations
when they are approached they are happy to give help and advice. It helped me to
them, so their time was not wasted and they were happy to help again.

Appendix B Relationships Diagram of Tables

Appendix C Add Week Function

Below is the Addweek function written in Visual Basic, which incremented the matc

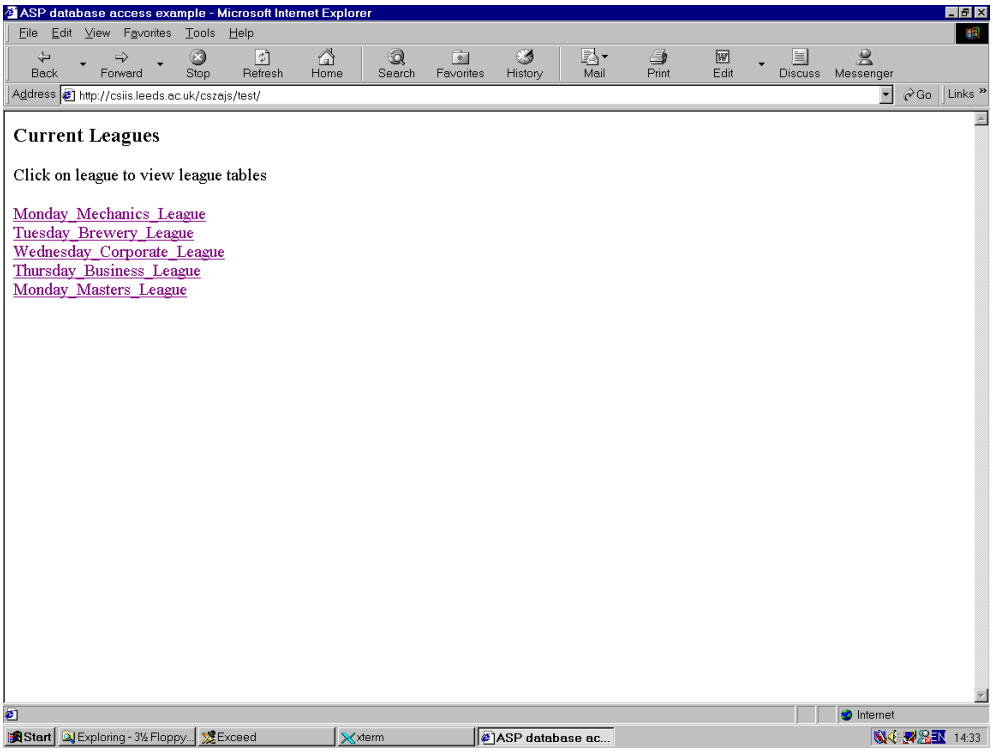
```
Function Addweek(d)

If DatePart("m",d)=1 Or DatePart("m",d)=3 Or DatePart("m",d)=5 Or DatePart("m",d)=7 Or
    DatePart("m",d)=8 Or DatePart("m", d)=10 Then
        If (DatePart("d", d) + 7) > 31 Then
                d = ((DatePart("d", d) + 7) - 31) & "/" & (DatePart("m", d) + 1) & "/" & DatePart("yyyy", d)
        Else
                d = (DatePart("d", d) + 7) & "/" & DatePart("m", d) & "/" & DatePart("yyyy", d)
        End If
Else

If DatePart("m",d) = 4 Or DatePart("m",d) = 6 Or DatePart("m",d) = 9 Or DatePart("m",d) = 11 Then
        If (DatePart("d", d) + 7) > 30 Then
                d = ((DatePart("d", d) + 7) - 30) & "/" & (DatePart("m", d) + 1) & "/" & DatePart("yyyy", d)
        Else
                d = (DatePart("d", d) + 7) & "/" & DatePart("m", d) & "/" & DatePart("yyyy", d)
        End If
Else

If DatePart("m", d) = 12 Then
        If (DatePart("d", d) + 7) > 31 Then
     d = ((DatePart("d", d) + 7) - 31) & "/" & ((DatePart("m", d) + 1) - 12) & "/" & (DatePart("yyyy", d) + 1)
        Else
     d = (DatePart("d", d) + 7) & "/" & DatePart("m", d) & "/" & DatePart("yyyy", d)
        End If
Else

If DatePart("m", d) = 2 Then
        If ((DatePart("yyyy",d)Mod 400)=0) Or (((DatePart("yyyy",d)Mod 100)<>0) And
                ((DatePart("yyyy",d)Mod 4) = 0)) Then
                If (DatePart("d", d) + 7) > 29 Then
                        d = ((DatePart("d", d) + 7) - 29) & "/" & (DatePart("m", d) + 1) & "/" &
                        DatePart("yyyy", d)
                Else
                        d = (DatePart("d", d) + 7) & "/" & DatePart("m", d) & "/" & DatePart("yyyy", d)
                End If
        Else
                If (DatePart("d", d) + 7) > 28 Then
                        d = ((DatePart("d", d) + 7) - 28) & "/" & (DatePart("m", d) + 1) & "/" &
                        DatePart("yyyy", d)
                Else
                        d = (DatePart("d", d) + 7) & "/" & DatePart("m", d) & "/" & DatePart("yyyy", d)
                End If
End If
End If
End If
End If
End If
End Function
```

Appendix Ð Example Reports

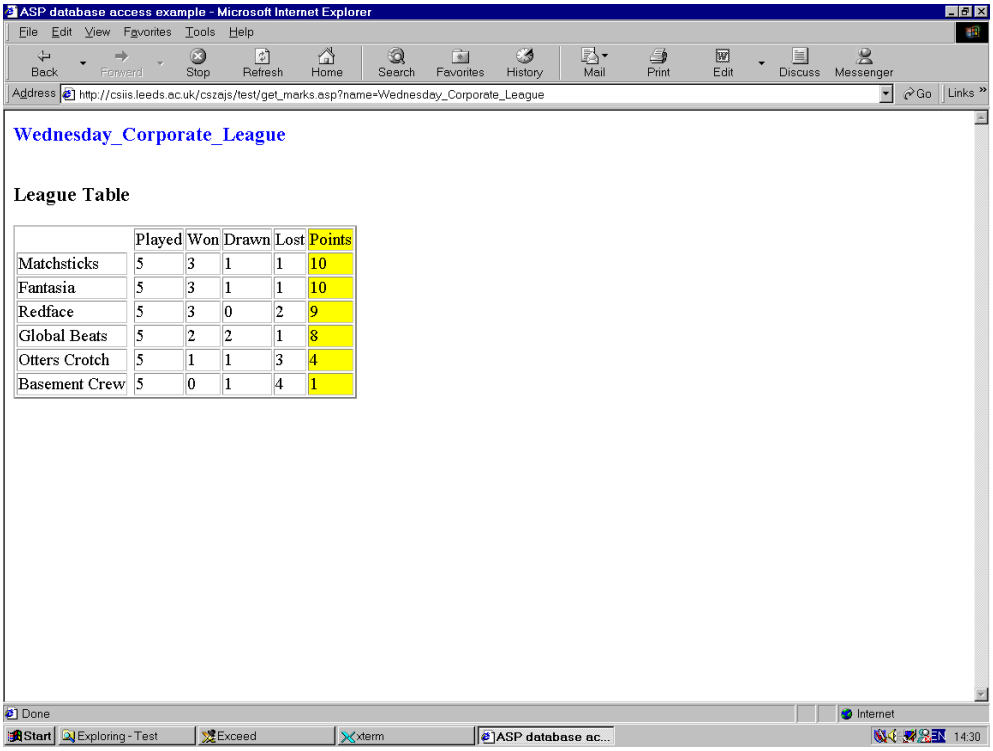
Appendix E ASP Files

Appendix F Web Page Screen Shots

The user is given a list of the teams currently running at Benyon Park:


ASP database access example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Messenger
Address http://csiis.leeds.ac.uk/cszajs/test/
Go Links
Current Leagues
Click on league to view league tables
Monday_Mechanics_League
Tuesday_Brewery_League
Wednesday_Corporate_League
Thursday_Business_League
Monday_Masters_League
Internet
Start Exploring - 3½ Floppy... Exceed xterm ASP database ac... 14:33

Here is the table returned when the ‘Wednesday Corporate League’ was clicked on:


ASP database access example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Messenger
Address http://csiis.leeds.ac.uk/cszajs/test/get_marks.asp?name=Wednesday_Corporate_League
Go Links
Wednesday_Corporate_League
League Table
Played Won Drawn Lost Points
Matchsticks 5 3 1 1 10
Fantasia 5 3 1 1 10
Redface 5 3 0 2 9
Global Beats 5 2 2 1 8
Otters Crotch 5 1 1 3 4
Basement Crew 5 0 1 4 1
Done
Internet
Start Exploring - Test Exceed xterm ASP database ac... 14:30